August 2018

Homotypic and Heterotypic Self-Assembly of Claudin Family of Tight Junction Proteins

Lisa Danielle Nguyen
Syracuse University

ABSTRACT

Tight junctions, found in all epithelial cells, are selective barriers that restrict the diffusion of ions and molecules within the paracellular space. They are important for maintaining cell polarity as well as for compartmentalization and establishing homeostasis within the human body. Tight junctions are comprised of complex protein assemblies. Within this protein assemble are a family of transmembrane proteins known as claudins that play a crucial role in establishing the tight junction network. Claudins are also influential in controlling the tight junction permeability. When mutations or malfunctioning occurs within a claudin gene, tight junction function is impaired. Disruption of their function is associated with a variety of human conditions, such as brain disease, deafness, renal failure, and various cancers. A deeper understanding of claudins and of their contribution towards tight junction's function will provide researchers additional insight as to how to eventually approach creating therapeutics to treat tight junction-related diseases.

In this thesis, homotypic and heterotypic *cis* self-assembly of claudin-claudin interactions were studied for both classic and non-classic claudins (-2, -4, -11, -14, -16, -18, -19, and -23). Homology modeling was utilized to generate structures for each of the eight claudins studied, which were then equilibrated and refined in a DOPC (1,2-Dioleoyl-sn-glycero-3-phosphocholine) lipid bilayer system. Consequently, self-assemble simulations were carried out to study the *cis* interaction in either homotypic or heterotypic fashion. Each self-assembly system contained 72 monomers and was simulated for 4 μs. Results showed aggregation of claudin monomers into strand-like assemblies, which were then analyzed through a dimer distribution and orientation analysis. Four dimer types (dimers A, B, C, D) were identified and

dimer populations were calculated in each of the claudin self-assembly systems. An additional new analysis method developed by a colleague and still in the refining phase is also introduced and discussed. Results for a single test system are discussed, but nonetheless provide an alternative means of analyzing dimers, representing them through various energy state profiles rather than of population density probabilities.

HOMOTYPIC AND HETEROTYPIC SELF-ASSEMBLY OF CLAUDIN FAMILY OF TIGHT JUNCTION PROTEINS

By

Lisa Nguyen

B.S., MCPHS University, 2016

Thesis

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Bioengineering

Syracuse University

August 2018

ACKNOWLEDGEMENTS

I would like to give my greatest thanks and appreciation to my advisor, Dr. Nangia. Thank you for giving me the opportunity to join your lab, as it has been one of the most wonderful things that I could have ever experienced. I have certainly learned a lot about molecular dynamics, tight junctions, and claudins; my world has very much expanded in thanks to you. You have been the kindest, encouraging, patient, and supportive advisor that I could have ever had throughout these two years. As I move onwards towards my future, the time I spent doing research here is something I will never forget.

I would also like to thank my fellow lab mates, Jerome, Nandhini, and Huilin. Thank you guys so much for helping me and mentoring me from the beginning to the end of my project. I am so glad to have met you all. I also could not have done it without you guys. Thank you for your easy to understand explanations to my many questions as well as your friendliness, encouragement, and kindness. It was always so pleasurable and fun to come into the lab to work because of you guys.

Finally, I would like to thank my dearest family and John. I am extremely grateful to have you in my life. Thank you for supporting me every step of the way throughout this journey, by telling me to work hard and to never give up. Thank you for constantly pushing me to finish what I started. I hope to succeed in life and make you guys proud.

TABLE OF CONTENTS

ABSTRACT i

ACKNOWLEDGEMENTS v

LIST OF FIGURES viii

LIST OF TABLES ix

I. INTRODUCTION 1

1.1 Tight Junctions 1

1.2 Claudins 5

1.3 Current Status 8

II. COMPUTATIONAL MODELING OF CLAUDINS AND TIGHT JUNCTIONS 13

2.1 Computational Modeling & Molecular Dynamics 13

2.2 Predicting and Generating 3D Structures & Homology Modeling 16

III. SELF-ASSEMBLY SYSTEM SETUP 20

3.1 Refining with Molecular Dynamics in Coarse Grain and Atomistic 20

3.2 Homotypic and Heterotypic Self-Assembly Setup 21

IV. ANALYSIS METHODS 24

4.1 Dimer Orientation Analysis Method 24

4.2 Mapping Energy Landscape Method 26

V. RESULTS & DISCUSSION 28

5.1 Analysis of Structural Models of Claudin Proteins 28

5.2 Self-Assembly Results & Analysis 29

5.3 Mapping Energy Landscape Method Results & Analysis 35

5.4 Comparison of the Two Methods 36

VI. CONCLUSION 38

VII. APPENDIX 41

7.1 Scripts to Sample Self-Assembly Population and Analyze Dimer Type Formation 41

7.1A ANGLE_RW.py script 41

7.1B 2D-KDE.py script 44

VIII. REFERENCES 47

IX. VITA 55

LIST OF FIGURES

FIGURE 1. Atomistic model versus coarse grain model of protein monomer …… 15

FIGURE 2. Homology modeling of claudin structures using YASARA …… 19

FIGURE 3. Homotypic and heterotypic self-assembly setup for claudin-claudin interaction using VMD software …… 22

FIGURE 4. 72 claudin proteins system setup for self-assembly interactions …… 23

FIGURE 5. Classification of dimer type conformations …… 24

FIGURE 6. Probability Density Function graph plot of dimer types …… 27

FIGURE 7. Analysis to evaluate protein structure and stability …… 29

FIGURE 8. Resulting self-assembly of homotypic classic claudins …… 30

FIGURE 9. Resulting self-assembly of homotypic non-classic claudins …… 30

FIGURE 10. Results from heterotypic self-assembly …… 31

FIGURE 11. PDF graphs of claudin-claudin self-assembly interactions …… 33

FIGURE 12. Sampling comparison of self-assemble versus mapping energy landscape plots …. 36

FIGURE 13. Mapping Energy Landscape energy analysis plot …… 37

LIST OF TABLES

TABLE 1. Claudin family of proteins and their associated properties …… 11-12

TABLE 2. Claudin-specific amino acid residue sequences sent to I-TASSER and QUARK servers to predict 3D structural models …… 17

TABLE 3. Dimer types identified from self-assembly organization …… 25

TABLE 4. Dimer type formation and population density comparison from self-assembly interactions …… 34

I. INTRODUCTION

1.1 Tight Junctions

In the human body; organ surfaces, hollow cavities, and blood vessels are lined by epithelial and endothelial tissue, which are important for compartmentalization of the various parts of the human body. The epithelial cells that make up the tissue play important roles for functions like secretion, selective absorption, protection, transcellular transport, and sensing. Between adjacent individual epithelial cells, there are distinct cell-cell junctions located paracellularly along the membrane, known as the epithelial junctional complex, consisting of tight junctions in the utmost apical position of the cell membrane, adherens junctions in the middle position, and desmosomes in the bottom position[1,2].

The nature of tight junctions and their molecular architecture has been more elusive than that of adherens junctions or desmosomes, so detailed research has been performed to enlighten others the mystery behind these junctions. Tight junctions were first discovered in 1963 by Farquhar and Palade through electron microscopic analyses of mammalian epithelial cells[2] and their discovery attracted much attention from scientific researchers due to their crucial role played in epithelial barrier function. Tight junctions are formed by two adjacent epithelial cells joining together to form a seal. They function as barriers that regulate the paracellular diffusion of ions and solutes into the cell, with selectivity based on the size and charge of the ions and solutes passing through. Tight junction paracellular transport is a completely passive transport, driven by electroosmotic gradient flow. There are two types of tight junctions: “leaky” and “tight”. “Leaky” tight junctions are located in areas of the body that need to transport large volumes of isosmotic fluid like the intestine, while “tight” tight junctions

are located in areas where high electroosmotic gradients are required, like the distal tubules and collecting ducts of the kidney[3]. Their function as barriers are significant in maintaining the state of homeostasis for the body. Disruption in their natural physiological function and state are associated with various human diseases.[4]

Under freeze-fracture electron microscopy, tight junctions have been observed to be a meshwork of fibers, formed by rows of transmembrane proteins,[5,6] predominantly claudins and the MARVEL domain proteins. Proteins considered as MARVEL domain proteins contain a four transmembrane-helix structure with cytoplasmic N- and C-terminal regions and are typically associated with cholesterol-rich membrane environments, such as occludin, MARVELD2 (tricellulin), and MARVELD3[4]. Both claudins and MARVEL domain proteins, under immunoelectron microscopy, demonstrate the ability to localize to tight junction strands. And in the case of non-tight junction forming cells, claudins have demonstrated that ability to prompt superficially similar tight junction strands to form and occludin demonstrate the ability to form short strand fragments[7-10]. Other transmembrane tight junction-associated components include lipolysis-stimulated lipoprotein receptors (angulins), BVES (blood vessel epicardial substance, CAR (coxsackievirus and adenovirus receptor), JAMs (junctional adhesion molecules), and a trispan protein[4, 11-16].

To function properly, tight junction transmembrane proteins need to interact with the cytosolic plaque. This is a complex protein network consisting predominantly of adaptor proteins that interact with the junctional membrane proteins' cytoplasmic domains in addition with microtubules and F-actin. There are different types of binding domains: three PDZ (PSD95, DlgA, ZO1 homology) domains, a SH3 (SRC homology 3) domain, and a GUK (guanylate kinase

homology) domain[4]. Each family of proteins interacts and binds with different domains. The first PDZ domain is for claudin binding, the third PDZ domain is where junctional adhesion molecules (JAMs) bind, and to the GUK domain is where occludin bind. The nature of these binding domain interactions results in tissue-specific functions for different types of tight junction transmembrane proteins and the types of tight junctions that resultingly form. Other plaque proteins include MAGI (membrane-associated guanylate kinase inverted) proteins, MUPP1 (multi-PDZ domain proteins), and a PATJ (PALS1-associated tight junction) protein[4], which are also important for establishing function for the tight junction transmembrane components.

Tight junctions function as barriers and they can form two barrier types: a paracellular barrier or an intramembrane barrier. The paracellular barrier regulates the transport of ions and solutes from between cells and through different body compartments while the intramembrane barrier acts as a fence to prevent the exchange of components within the membrane of the basolateral to the apical cell surface domains[4].

Regarding the permeability of the paracellular barrier, it is affected by two factors: solute charge and size. In a charge-selective paracellular pathway, ions and small charged molecules can cross through the tight junction, through pores that are estimated to be ~4-8 Å wide in diameter[17-19]. In comparison, a size-selective pathway allows larger solutes and molecules to pass through, with their size being around ~30-60 Å[17,19]. To determine the degree of ion permeability in the charge-selective pathway and how likely ions are to diffuse across the paracellular cleft, transepithelial resistance (TER), an instantaneous measurement of the electrical resistance within the pathway, is measured. As for large molecules in size-selective

diffusion, diffusion is slow and measured over longer periods of time with tracers. Research progress has been made to uncover more of the underlying mechanism of charge-selective ion permeability while larger molecule diffusion is less understood. From the progress in research made regarding charge-selective ion permeability, claudin proteins appear to be the biggest key players in influencing ion permeability across the paracellular space.

As for the intramembrane barrier of the tight junctions, its fence function has been studied by experts by using fluorescent lipid probes and lipids, where the lipids exhibited what was a diffusion barrier since they were observed to be not intermixing with the other lipids between the apical and basolateral sides within the outer part of the plasma membrane. The intramembrane barrier of the tight junctions is important for maintaining a state of epithelial polarity. However, this role is somewhat complicated, as some cells with mutated intramembrane barriers can still polarize[4]. Further research needs to be done to understand and elucidate how the intramembrane barrier contributes to an epithelial cell's physiological function.

The process of tight junction assembly is important and ultimately affects overall tight junction function, including that of polarity and of tissue-specificity. The assembly of tight junctions involves a ZO1-α-catenin complex to couple tight and adherens junctions [20,21], nectins to recruit JAMA (junctional adhesion molecule-A) [22], and signaling mechanisms like protein kinase C (PKC), PKA, AMP-activated protein kinase (AMPK), protein phosphatases, small GTPases, and heterotrimeric GTPases [4, 23-27] once they maturely form. Other signaling mechanisms are involved as well, but how these different mechanisms work with one another

and guide the physiological function of cells needs to be further studied for better understanding.

1.2 Claudins

Of all the components involved with tight junctions, claudins are the key players responsible for the selective size, charge, and conductance properties of tight junctions[28,29]. The regulation of both claudins and tight junctions for barrier function is important to the wellbeing of the human condition, as disease onsets of various cancers, vision abnormalities, hearing loss or deafness, respiratory infections, gastrointestinal inflammation and diseases, and renal failure[30] are associated problems with claudin gene mutations that lead to a loss in regular tight junction function.

Shoichiro Tsukita and colleagues in 1998 discovered claudin proteins and connected their essential contribution to tight junctions[31], as they were the first set of proteins demonstrated to have tight junction-forming activity[32]. Protein overexpression, knockdown, and knockout experimental tests have been performed on these proteins and subsequent results indicated that claudins were the most significant components in the tight junction to affect its flux control[33].

Claudins comprise a 27-membered family of transmembrane proteins. Within the 27 members of the claudin family of proteins, two main groups have been established to categorize them: classic (claudins -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -14, -15, -17, and -19) and non-classic (claudins -11, -12, -13, -16, -18, -20, -21, -22, -23, -24, -25, -26, -27)[34,35]. Classic claudins share high protein sequence similarity with one another and their extracellular loop

amino acid residues have similar lengths and are very similarly commonly present throughout. All other claudins that do not share high degrees of sequence similarity with one another are categorized thus as non-classic claudins. Structurally, all claudins, both classic and non-classic, consist of four domain regions embedded in the cell membrane, two extracellular loops located outside of the cell, and an N-terminus and C-terminus located within the cytoplasm of a cell. The N-terminal tail end of claudins is relatively short, consisting of around seven amino acids in length. Meanwhile, the C-terminal tail is longer, around 25-55 amino acids, and is responsible for influencing the degree of stability of claudins and their targeting to tight junctions [36]. The first extracellular loop (ECL) contains 40-50 amino acids and the second loop contains around 15-30 amino acids[33,35]. The first ECL loop contains charged amino acids and influences the degree of selectivity of paracellular transport based on charge while the second ECL loop influences the degree of selectivity based on ion particles sizes to be passing through the paracellular cleft[35]. Other characteristics include disulfide bond presence, a signature GLW amino acid sequence, an ion-binding site on the first ECL, and palmitoylation sites[33].

Various claudin subtypes are found in the makeup of the tight junction strands, interacting via *cis* and *trans* to form them. Individual claudin proteins first self-assemble along the same membrane in *cis* and then interact in *trans* with other self-assembled claudin proteins on an opposing membrane to form the tight junctions[37]. When claudins come together, both non-classic and classic, and interact in both *cis* and *trans* to form the tight junctions, these interactions can be of both homotypic (involving the same claudin proteins) or heterotypic (involving different claudin proteins) nature. Claudins can form both heterotypic and homotypic tight junctions, but their affinity for one another is dependent on the claudin subtypes

involved[32]. In a study of cocultures of L cells expressing claudin -1, -2, and -3, combinations of claudin-1/-3 and claudin-2/-3 showed the formation of tight junction strands, but not the case for claudin-1/-2[38]. In another study done with HeLa cells, heterotypic interactions of claudin-1/-3 and claudin-3/-5 underwent assembly, but the same case was not observed for heterotypic interactions of claudin-4/-5, claudin-1/-4, and claudin-3/-4 [39]. As there are better partners for heterotypic interactions, there are also special exceptions where certain claudin subtypes cannot interact homotypically with one another, as is the case with claudin-16. When interacting homotypically, claudin-16 does not show any association with itself to assemble into tight junction strands. However, when incorporated with claudin -19, heteromeric interaction and tight junction strand formation is observed[40].

As claudin protein expression is tissue-specific and each claudin subtype has individual attributes, the amalgamation of these determinants directs the barrier or pore properties of each cell type's paracellular pathway[32]. Barrier-forming claudins increase transepithelial resistance and decrease solute permeability. Meanwhile, pore-forming claudins decrease transepithelial resistance and increase solute permeability[33]. While tight junctions function as epithelial barriers to restrict diffusion of solutes through the paracellular cleft, their barrier and channel properties and degree of solute permeability are dependent on the nature of claudin expression and interaction. Certain claudins have been known to be cation or anion-selective and form barrier or channel/pore tight junctions, predominantly forming one tight junction property configuration over the other based on whatever current ions are present. Previous research in the field has established and determined the barrier and pore-forming properties of the following claudins: claudins -2, -10, and -15 act as cation-selective pore-forming claudins,

claudins -10 and -17 act as anion-selective pore-forming claudins, claudins -7 and -19 form cation-selective barriers, and claudins -1, -3, -4, -5, -6, -8, -9, -11, -14, and -18 form anion-selective tight junction barriers[33]. Table 1 lists and describes the various properties of all the claudin proteins such as selectivity, protein length, classification, etc.

1.3 Current Status

Though research has progressed significantly since 1963 when tight junctions were first discovered, their true structure remains to be established[4]. Experimental approaches have looked at tight junction assembly mechanisms and different claudin proteins interactions with respect to how they form tight junction strand networks. One such experiment, performed by Milatz et al. looked at the isoforms of claudin-10 and claudin-10 chimeras and observed how they interacted with each other to form tight junction-like strands[41]. Taking three claudin-10 isoforms and two claudin-10 chimeras to use in their experiment, the researchers demonstrated that the proteins were able to interact in *cis* with each other using FRET (Forster/fluorescence resonance energy transfer) and then further interact in *trans* to integrate into tight junction-like strands by viewing the results under freeze-fracture electron microscopy. One redundancy in experimental approaches toward studying one protein is that basic conditions of experimental setups may mean that a protein being currently researched is only important under certain conditions or in a specific type of cell or system[4]. Compared to experimental means of studying proteins like claudins, computational methods provide a means to look at an entire protein system in a less biased way, as computational methods can incorporate many types of cells and systems to study, sampling more varied conditions.

My colleagues in the Nangia Lab work with computational tools and methods to study protein and protein systems such as claudins. Previous research done in the lab includes studying and reporting the results of a *cis* natured type of self-assembly, a non-bonded energy directed organization of system components from a disordered state to one of equilibrium, of classical claudins -1,-2,-3,-4,-5,-15, and -19 and a *trans* type self-assembly for claudin-5. Their self-assembly simulations showed that claudin proteins in *cis* self-assembled into dimeric and trimeric interfaces that can form even larger strands. Additionally, they looked into the pore structure of these classic claudins and revealed more about their selectivity in terms of individually as well as how they contribute to overall tight junction function[42-44].

Self-assembly resulted in strand formation of claudin proteins, with the basis of each claudin protein monomer forming dimeric interfaces with one another. Four dimeric interfaces have been identified so far through previous work done by the Nangia Lab[42-44], with each dimeric interface indicating the orientation to which claudin monomers aggregated close together individually as they formed strands. These orientation classifications: Dimer A, B, C, and D, indicate the patterns that claudin interactions undergo the most in a *cis* self-assembly. This organization of the dimer types is pertinent information that can be used to hypothesize how claudins form the barrier and pore-types of structures that they do in *trans* interactions.

In this thesis, the resulting self-assembly organization of classic and non-classic claudins -2, -4, -11, -14, -16, -18, -19, and -23 in *cis* are observed and analyzed to see what type of dimer orientations results, just like previous work done by the lab for classic claudins. Interactions of both homotypic and heterotypic conditions will be done for these claudins. The homotypic interactions will consist of claudin-2/-2, claudin -4/-4, claudin -11/-11, claudin -14/-14, claudin -

16/-16, claudin -18/-18, claudin -19/-19, and claudin -23/-23. In addition, heterotypic interactions of claudin -2/23 and claudin -4/23 will be simulated and compared to their homotypic self-assembly counterparts. Researching both non-classic and classic claudins in homotypic and heterotypic interactions and comparing the two to each other can give insight about any patterns or differences that may occur between a similar group (classic claudins) versus a group that is more dissimilar (non-classic claudins). Their individual differences in sequence and overall structure will give different strand formation results, leading to different dimer types, indicating different properties for stability and function for a claudin protein.

Furthermore, limitations of the current self-assembly method will be discussed and a brief introduction to a new method involving energetics analysis, currently in the testing phase for analyzing claudin-claudin interactions and their resulting dimer confirmations, developed by my colleague in the lab, will be briefly covered.

TABLE 1. *Claudin family of proteins and their associated properties.* Table listing all claudins family members and information regarding expression, classification, and associated pathologies with each respective claudin.

Name	Amino Acid Length	Tissue Expression [33,34]	Associated Pathologies [33,34]	Pore or Barrier	Non-Classic or Classic
Claudin-1	211 [58]	Epidermis, distal nephron, gallbladder, inner ear, ovary	Skin diseases, neonatal sclerosing cholangitis, hepatitis C virus infection	Anion Barrier	Classic
Claudin-2	230 [58]	Intestinal crypts and proximal tubule of kidney	Crohn's disease	Cation Pore	Classic
Claudin-3	220 [58]	Respiratory, urinary, and GI tract; salivary and mammary glands; blood brain and blood-testis barrier	Cancer (ovarian, prostate, colorectal, and breast), autoimmune encephalomyelitis, and gliobastoma tumors	Anion Barrier	Classic
Claudin-4	209 [58]	Gallbladder, collecting ducts of kidney, lung	Various cancers (ovarian, prostate, breast, colon)	Anion Barrier	Classic
Claudin-5	218 [58]	Brain, ovaries, colon	Crohn's disease, DiGeroge/Velo cardiofacial syndrome, hyperplastic vessels in glioblastoma, pancreatic cancer	Anion Barrier	Classic
Claudin-6	220 [58]	Embryonic epithelia, neonatal kidney,		Anion Barrier	Classic
Claudin-7	211 [58]	Lung, kidney, intestine, colon	Cancer (stomach, breast, head, neck)	Cation Barrier/Anion Pore	Classic
Claudin-8	225 [58]	Lung, kidney, intestine, colon	Crohn's disease	Anion Barrier	Classic
Claudin-9	217 [58]	Inner ear	Deafness	Anion Barrier	Classic
Claudin-10	228 [58]	Neonatal kidney, inner ear, kidney nephron, salivary gland, epididymis		Cation Pore/Anion Pore	Classic
Claudin-11	207 [58]	Neuron myelin sheaths, Sertoli cells	Hearing abnormalities	Anion Barrier	Non-Classic
Claudin-12	244 [58]	Inner ear, brain, intestine, colon			Non-Classic

TABLE 1. (CONTINUED)

Name	Amino Acid Length	Tissue Expression [33,34]	Associated Pathologies [33,34]	Pore or Barrier	Non-Classic or Classic
Claudin-14	239 [58]	Inner ear, thick ascending limb of kidney	Nonsyndromic deafness, cochlear hair cell degeneration	Anion Barrier	Classic
Claudin-15	228 [58]	Kidney, intestine, colon		Cation Pore	Classic
Claudin-16	305 [58]	Thick ascending loop of Henle, nephron	Familial hypomagnesemia with hypercalciuria and nephroclcinosis (FHHNC)	Cation Pore	Non-Classic
Claudin-17	224 [58]	Kidney, taste receptors, brain		Anion Pore	Classic
Claudin-18	261[58]	Stomach, lung, and inner ear	Gastric cancer	Anion Barrier	Non-Classic
Claudin-19	224 [58]	Renal tubules, retina, and myelin sheaths of neurons	FHHNC, visual impairment	Cation Barrier	Classic
Claudin-20	219 [58]	Skin, mRNA	Chondrosarcoma, brain and liver cancer		Non-Classic
Claudin-21	229 [33]	DNA, intestine, stomach, liver, kidney			Non-Classic
Claudin-22	220 [58]	Trachea, mRNA	Breast cancer, astrocytoma		Non-Classic
Claudin-23	292 [58]	mRNA, colon, stomach, placenta, skin	Intestinal gastric and colon cancer, atopic dermatitis	--	Non-Classic
Claudin-24	220 [58]	DNA, intestine, stomach, kidney, heart			Non-Classic
Claudin-25	229 [58]	Intestine, stomach, liver, kidney, heart, brain			Non-Classic
Claudin-26	223 [33]	Intestine, brain, eye			Non-Classic
Claudin-27	208 [33]	Intestine, liver			Non-Classic

II. COMPUTATIONAL MODELING OF CLAUDINS AND TIGHT JUNCTIONS

2.1 Computational Modeling & Molecular Dynamics

For researching and uncovering the nature of claudins and how they contribute to the overall tight junction formation and function, computational modeling is utilized. It is a modeling method that uses computers to study and simulate complex real-life phenomenon using the principles of physics, mathematics, and computer science. Using refined models and powerful software packages and computer resources, theoretical computational studies give representations and results of complex systems, revealing much more data and information than experimental methods are limited in giving. Results achieved through computational means are processed more quickly and can be repeated in a shorter time-scale if necessary to give accurate results, compared to their experimental equivalents.

This research uses computational modeling to generate the 3D structures of claudin proteins and organize them into the desired system to simulate and observe their patterns of self-assembly, an important biomolecular mechanism that involves the autonomous organization of components into patterns or structures[45]. Aside from observing whether claudins in their self-assembly lead to the formation of a strand-like network, as seen in tight junctions, observed will be the differences present in self-assembly for various claudin types. One of the tools available and used for this research to study claudins and their self-assembly is molecular dynamics (MD). MD is a computer simulation method that studies the physical movements of atoms and molecules[46]. Based on Newton's second law of motion, MD takes a static model of a biomolecule system and calculates the molecular forces acting on each atom, nonbonded and bonded energies, and let the atoms and molecules within the system interact

with one another over a fixed period of time to observe the dynamic evolution of the movement of atoms in the system over time.

MD is run under the guidelines of force fields, equations set to parameterize the atomic and molecular components of the system, dictating the conditions of MD operations to calculate overall potential energies of movement of system components. Two types of representations of a molecule are atomistic and coarse-grain models, shown in Figure 1, with each type of model represented by a different type of force field. An atomistic model would utilize a force field such as CHARMM[47] where every type of atom within the molecular system is accounted for. Meanwhile, a coarse-grained model utilizes a MARTINI force field[48], which maps every number of four heavy atoms as one larger bead, thereby simplifying the overall representation of atoms in the system. When simulating a molecular system in an all-atom scale of representation, results are detailed but limitations arise with large system sizes, as the level of detail in an all-atom scale of representation leads to requirements of long simulation times to completely calculate all potential energy and movement within the large system. Coarse-grain model simulations, on the other hand, are more simplified in their representation due to the mapping of heavy atoms into beads and thus make for more rapid calculating of energy potentials than compared to atomistic models. It is better to run simulations in coarse-grain representation due to the higher efficiency when computing for large molecular system sizes.

Certain packages are available for users to use to run molecular dynamics, such as GROMACS (GROningen MAchine for Chemical Simulations)[49]. This package comes with

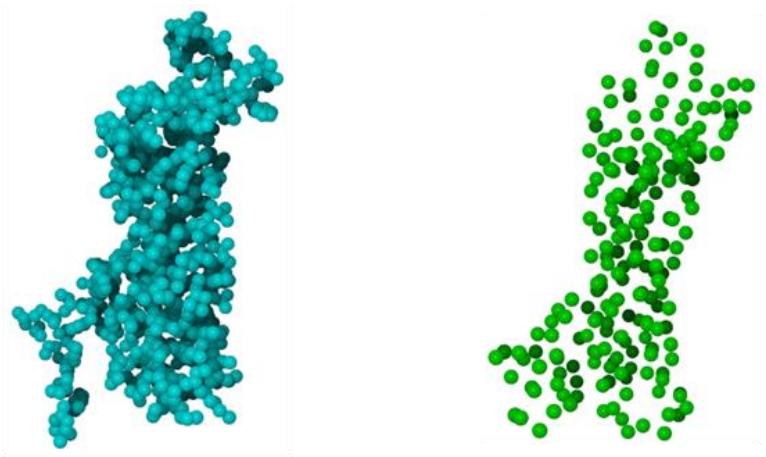


FIGURE 1. *Atomistic model versus coarse grain model of protein monomer.* Depicted is two different representations of a claudin protein, the left is in atomistic (cyan) and depicts all atoms of the protein including hydrogens while the right side is in coarse-grain (green) and represents *n* number of atoms per 1 bead (1 bead=1 green dot).

some necessary tools to create a well-prepared system to run an accurate MD simulation of the desired biomolecule. The preparation steps include running energy minimization, equilibration, NVT, and NPT before running the production MD simulation to get as accurate a result of how the molecule may interact over time within the specified system. Energy minimization relaxes the structure to ensure that there are no steric clashes or inappropriate geometry. Equilibration takes surrounding molecules like ions or water in a system and optimizes them around the target molecule itself. The temperature of the system is brought up to a specified value, known as running an NVT or isothermal-isochoric ensemble, followed by an NPT or isothermal-isobaric ensemble, where a pressure is applied to the system until a stable density is reached. Once the system is well-equilibrated where the temperature and pressure are adequate, a production

MD run is executed, simulating the movements of molecules to observe the dynamics of the system from initial time period to a specified simulation period.

2.2 Predicting and Generating 3D Structures & Homology Modeling

Since many of the 3D structures of claudins are not entirely known, the use of servers like I-TASSER[50], QUARK[51], FG-MD[52], locPREFMD[53], RAPTOR-X[54], RCD+[55], and PPM[56] are utilized to predict, generate, and refine the resulting 3D structures of this work's target claudin proteins. Once complete, these structures are compiled and undergo homology modeling, using the YASARA[57] program, to predict the most accurate individual 3D structures as possible. Only then after these preparation steps, can the claudin models then be further used to set up the system for self-assembly.

I-TASSER (Iterative Threading ASSEmbler Refinement) is an online server run by Zhang Lab of University of Michigan that predicts 3D structures of proteins based on their amino acid sequences. It predicts the protein's secondary structure, solvent accessibility, and normalized B-factor (a value indicating the stability of the residues in the protein) of the protein and generates different 3D structural models, predicting how each one would look. The resulting models have c-score values, indicating the amount of confidence that the server had in the resulting models. C-score values range from [-5,2] but a c-score value greater than -1.5 "indicates a model of correct global topology"[50]. Predicted models that had a good c-score value range from -1 to 1 were selected and used in the next step of homology modeling by YASARA.

Next, QUARK, also run by Zhang Lab, was used. It predicts 3D protein structure as well, with the exception that their models are based off free-modeling through replica-exchange Monte-Carlo simulation instead of established homologous templates[51]. For generating 3D models on both the I-TASSER and QUARK servers, protein FASTA sequences for claudin -11 (Entry ID: *O75508*), -14 (Entry ID: *O95500*), -16 (Entry ID: *Q9Y5I7*), -18 (Entry ID: *P56856*), -19 (Entry ID: *Q8N6F1*), and -23 (Entry ID: *Q96B33*) were taken from the UniProt database[58] and uploaded onto each of the servers. Generating claudin -2 and -4 structures from their FASTA sequences were not included, as complete homology models were already readily available to use due in thanks to previous work done by a colleague, Flaviyan J. Irudayanathan. The more known and stable regions of the claudin sequences were sent to I-TASSER and the more unstable and disordered regions, mainly the tail ends of the claudin protein, were sent to the QUARK server. Table 2 describes which part of the claudin's amino acid sequence was sent to which server.

NAME	I-TASSER	QUARK
CLDN-11	1-190	181-207*
CLDN-14	1-190	191-239
CLDN-16	61-260	1-60, 261-305
CLDN-18	1-200	201-261
CLDN-19	1-190	191-223
CLDN-23	1-190	191-292

TABLE 2. *Claudin-specific amino acid residue sequences sent to I-TASSER and QUARK servers to predict 3D structural models.* Stable protein sequence regions of each claudin sent to I-TASSER using a multiple threading approach while disordered regions sent to QUARK using replica-exchange Monte Carlo simulation.

*residue starts from 181 instead of 191 due to 20aa minimum for QUARK system

** claudin -2 and -4 not included, as homology models have already been generated

Additional servers and tools used included the FG-MD, locPREFMD server, and the RAPTOR-X Contact Prediction tool. FG-MD (Fragment-Guided Molecular Dynamics) uses algorithms to refine initial protein models to look more like their native structures, reshaping them into appropriate geometries that contain no steric clashes, with improved torsion angles and hydrogen-binding[52]. The locPREFMD (*local* Protein structure REFinement via Molecular Dynamics) server by Feig lab further improves upon stereochemistry of atoms in the protein molecule[53], and the RAPTOR-X Contact Prediction tool looks at protein model residue contacts and evaluates the quality of that contact prediction through confidence scores[54]. In addition to these servers being used to generate and refine the 3D claudin structures, the RCD+ (Random Coordinate Descent) server was specifically used on the loop sequence of claudin-18 to refine it, using an algorithm involving geometric algebra to update its loop arrangement and to enhance the random selection of bonds[55].

Once the servers finished their jobs and completed predicting and refining the protein sequences, their files were compiled into YASARA (Yet Another Scientific Artificial Reality Application) to undergo homology modeling, the process of using 3D crystal structure models of other known proteins as templates due to known sequence and structural similarity, to generate accurate target protein models in 3D. Not only were the files from the servers used as templates, but the recently discovered and known crystal structures of claudin -4 (PDB:5B2G)[59], claudin-15 (PDB:4P79)[60], and claudin-19 (PDB:3X29)[61] were used as templates as well. Oligomerization state option was set to one and the number of templates option was set to a minimum of five templates. Once homology models for claudins -2, -4, -11, -14, -16, -

18, -19, and -23 were completely generated, as shown in Figure 2, each claudin was sent to the PPM server of the Orientations of Proteins in Membranes (OPM) database to find their correct orientation within a membrane, as this server orients transmembrane and peripheral proteins properly into a membrane by calculating its relative position and orientation within a membrane with consideration for the properties of hydrophobic thickness, tilt angle, and transfer free energy[56].

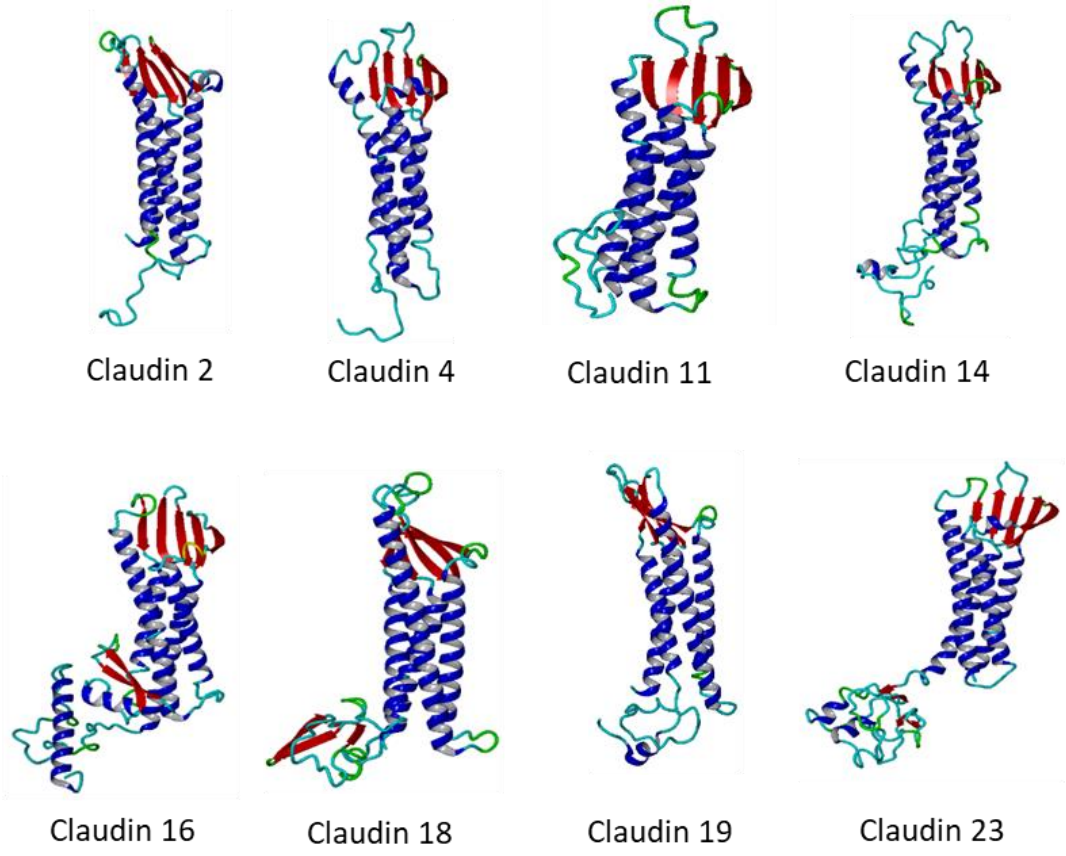


FIGURE 2. *Homology modeling of claudin structures using YASARA.* Resulting 3D homology-modeled structures of claudin monomer proteins. Transmembrane helical domains are colored in blue. Beta sheets are colored in red.

III. SELF-ASSEMBLY SYSTEM SETUP

3.1 Refining with Molecular Dynamics in Coarse Grain and Atomistic

Once homology modeling was complete, claudin structures were prepared for MD simulation. Claudins -2, -4, -11, -14, -16, -18, -19, and -23 were individually uploaded into the martini bilayer maker from CHARMM-GUI, using the elnedyn martini force field, to input itself into a coarse-grained lipid membrane system[47]. The system size was set to 75 Å in the X and Y direction, with a 1:1 upper leaflet and lower leaflet ratio of DOPC (1,2-Dioleoyl-sn-glycero-3-phosphocholine) lipid, generated with 0.15 M of NaCl ions and put in a water box. The temperature was increased and set to 310.15 K for isothermal-isochoric (NVT) and isothermal-isobaric (NPT) ensemble.

The coarse-grained system initialized first with an energy minimization run for 50 ns with no position restraints. After energy minimization, a series of equilibration steps was run for 500 ns with a position restraint on the lipid bilayer and at a temperature of 310.15 K for NVT and NPT ensemble. Following equilibration, a dynamics production run was executed without position restraints for a simulation time of 2 μs.

Once the coarse-grained production run finished simulating, the final file generated from the production run is input into CHARMM-GUI again to be reverse-mapped into an all-atom system in the CHARMM forcefield using the *backwards.py* script[47,62]. Using the input all-atom converter martini maker input generator, terminal group patching and disulfide bonds are selected and identified for the program to include. Then, ions and a water box are built around the membrane components.

Like the coarse-grained system, a simulation was run starting with energy minimization, then several steps of equilibration, followed by a production run. Energy minimization was run on the atomistic system with no position restraints for 5 ns. After minimization, an equilibration run was done for 1 ns of time at a temperature of 310.15 K with hydrogen bond constraints according to LINCS algorithm. Finally, a production run followed for 100 ns with Nose-Hoover temperature coupling at 310.15K with hydrogen bond constraints from LINCS constraint algorithm. Once the atomistic simulation completed production, the structure was released, and a cluster analysis was performed, taking the first cluster of each claudin to use for their respective setups for self-assembly.

3.2 Homotypic and Heterotypic Self-Assembly Setup

For both homotypic and heterotypic claudin-claudin self-assembly setups, two claudin monomer proteins, retrieved from the cluster analysis, are put in a grid with 5 nm diagonal distance apart. For the homotypic claudin-claudin setups, two of the same claudins were put in place together; heterotypic claudin-claudin setups involved two different claudins put together. See Figure 3. Homotypic self-assembly setups were done for claudin -2/-2, claudin -4/-4, claudin -11/-11, claudin -14/-14, claudin -16/-16, claudin -18/-18, claudin -19/-19, and claudin -23/-23. As for heterotypic setups, they were claudin -2/-23 and claudin -4/-23.

For every claudin-claudin system that was set up, the *martinize.py* python script[63] was used to map the system back into coarse-grained beads instead of individual atoms. Secondary structure files were also retrieved and run with the *martinize*.py script to generate an .itp file for later use with the topology file. Following the previous action, the *insane.py* script[64] was

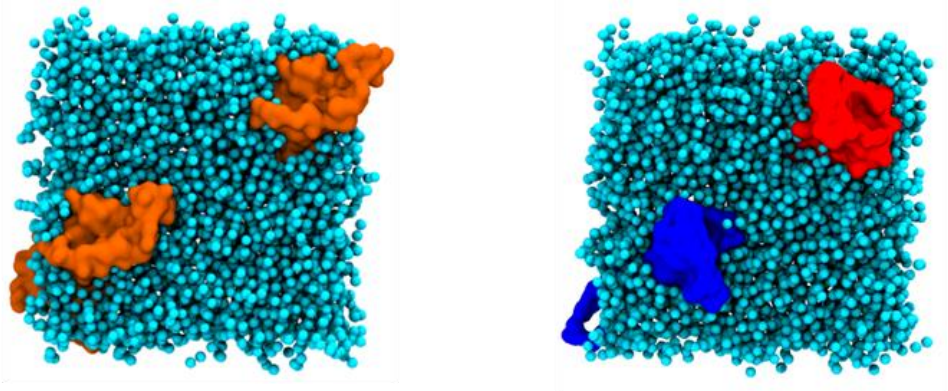
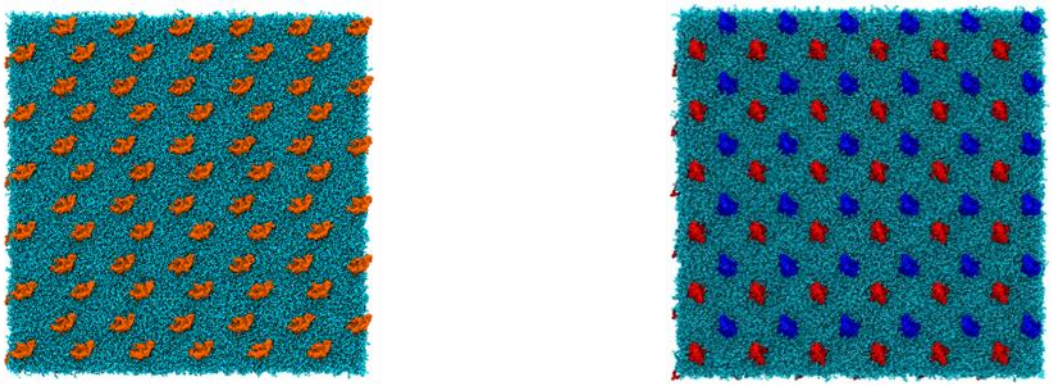


FIGURE 3. *Homotypic and heterotypic self-assembly setup for claudin-claudin interaction using VMD software.* Homotypic interactions for claudin -2/-2, -4/-4, -11/-11, -14/-14, -16/-16, -18/-18, -19/-19, and -23/-23 involve the same two proteins interacting with each other (left) while heterotypic claudin self-assembly (right) involve two different proteins, as is the setup for claudin -2/-23 & claudin-4/-23.

used to generate a box around the claudin-claudin grid and surround it in a lipid bilayer and fill it with ions and water. VMD (Visual Molecular Dynamics) software[65] was used to visualize the system.

Once all claudins were properly mapped and orientated in the lipid bilayer membrane, using the *gmx genconf*[49] command available through the GROMACS package, the two monomer grid was multiplied and scaled up to a larger grid size containing 72 monomer proteins. Homotypic systems contained 72 of the same claudin monomers while heterotypic systems contained 36 of one type of claudin and 36 of another. See Figure 4. The initial box size increased from (10.1, 10.1, 9.8) to (61.5, 61.5, 9.7) in the *xyz* direction. Energy minimization and a series of equilibration steps are run again to prepare the system for the self-assembly

production run. Once they completed, the production run was set to run for a period of 4 μs in triplicate.



Homotypic Heterotypic

FIGURE 4. *72 claudin proteins system setup for self-assembly interactions.* System size has increased from 2 protein system to 72 protein system to include more claudin sample size for self-assembly simulation in both homotypic (left) for claudin -2/-2, -4/-4, -11/-11, -16/-16, -18/-18, -19/-19, and -23/-23 and heterotypic (right) for claudin -2/-23 and -4/-23.

IV. ANALYSIS METHODS

4.1 Dimer Orientation Analysis Method

Previously developed by my colleagues at the Nangia Lab, they have observed the patterns of claudin-claudin self-assembly and their results led them to identify their organization by classifying the types of interactions they see as dimer types: dimer A, B, C, and D[42-44]. See Figure 5. Dimers are defined by the degree of angle rotation undergone between the transmembrane regions of two claudin proteins. Each dimer is classified by a unique degree of rotation with respect to one another. See Table 3.

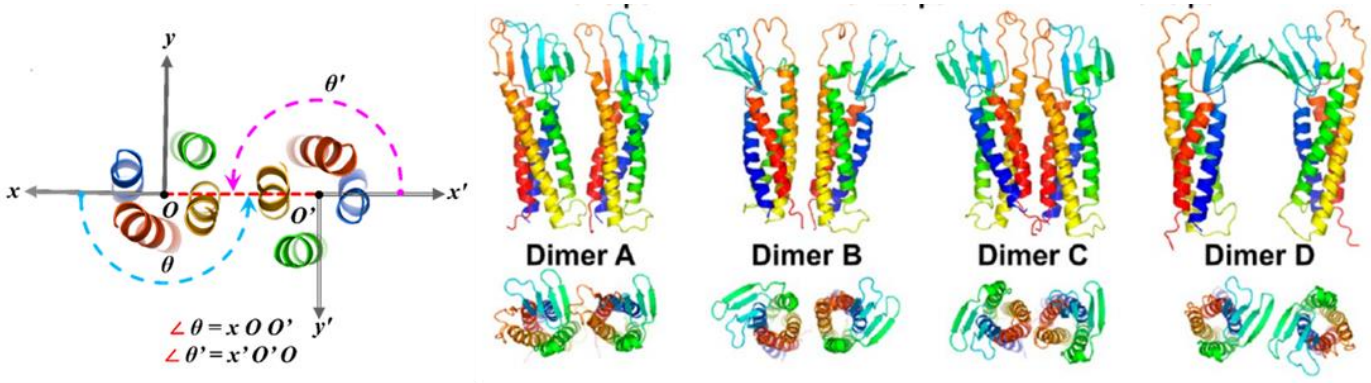


FIGURE 5. *Classificiation of dimer type conformations.* 4 groups of dimers identified and named Dimer A,B,C, and D. Based on the rotation of transmembrane helices of one protein with respect to another. Transmembrane helices 1 to 4 are colored each in red, yellow, green, and blue.

Observed through previous research, claudin-claudin self-assembly leads to strand formation. Therefore, scripts were generated to further analyze the meaning behind the strand formation and the types of individual units, or dimers, leading to their strand formation. The ANGLE_RW.py and 2D-KDE.py scripts were created and used to analyze the angle and dimer types that formed from homotypic and heterotypic self-assembly. See Appendix 7.1 for more

Dimer	Location	Location
Dimer A	[π/2, 3π/2]	[3π/2,π/2]
Dimer B	[π, π]	[π, π]
Dimer C	[π/2. π/2]	[π/2, π/2]
Dimer D	[0,2π]	[2π,2π]

TABLE 3. *Dimer types identified from self-assembly organization.* Location of the various dimer types based on their degree of angle rotation with angle space.

detailed information into the strings and commands for both of the scripts used. Generating plots with the ANGLE_RW.py and 2D-KDE.py scripts give the following figure, Figure 6, that visualizes the types of dimer conformations as well as how populous that dimer type is. It shows their distribution in the form of a graph based on kernel density estimation, a sampling tool used in statistics that looks at the population or conformation of a protein most likely to appear. Based on the kernel density estimation, the graph generated is a probability density function (PDF) type of graph, where the density of a random variable is measured, indicating the relative likelihood that the random variable would be equivalent to the whole sample. Areas filled with more of the blue color correspond to low probability for dimer population, while more of the red color correspond to higher probability. Additionally, within the plot, boxes lettered A, B, C, and D correspond to the regions where the respective dimer types are spatially located.

4.2 Mapping Energy Landscape Method

An alternative means of analyzing the self-assembly dimer conformation types was developed by my colleague at the Nangia Lab, Nandhini Rajagopal. This new method, termed the Mapping Energy Landscape method, analyzes dimers in terms of energetics, rather than based on the population like the previous method. While this method is currently in testing phase, it's results have provided additional insight into the nature of understanding claudin-claudin interactions and their resulting dimer formation.

Instead of setting up the system with 72 claudins in a grid with a large box size, this method takes 2 claudin monomers and puts them together in a box, filling the box with the same components as the self-assembly method: lipids, water, and ions. Taking the two claudin proteins, a number of random orientations, each one different, are generated. Following this step, a short simulation is run to energy minimize and equilibrate the system, letting the claudin proteins interact and orient however they like. At the end of the simulation time, the claudin proteins are able to sample the entire angle space and the resulting energies associated with each angle orientation are plotted on a graph for visual representation. Results should display a profile of various energies associated with different angle conformations. The conformations lowest in energy values should correspond to the most stable of dimers.

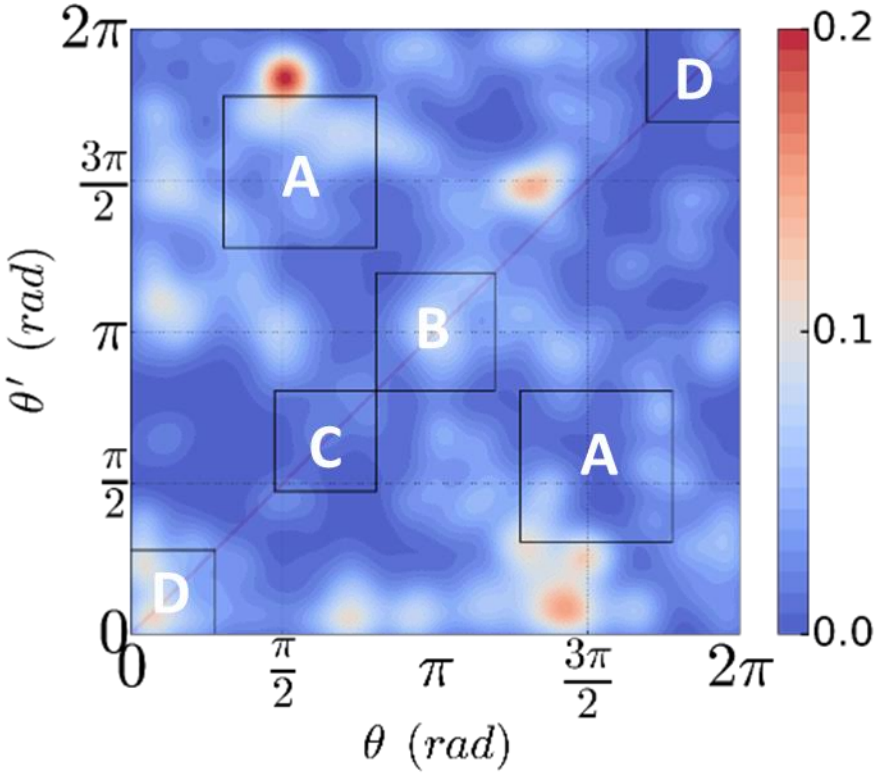


FIGURE 6. *Probability Density Function graph plot of dimer types.* Graph based on kernel density estimation to show the locations of dimer types A,B,C, and D. Density is shown as well; red color shows dense population of dimer type while blue shows none.

V. RESULTS AND DISCUSSION

5.1 Analysis of Structural Models of Claudin Proteins

Before proceeding onwards with self-assembly simulations, the generated claudin monomer structures were analyzed through various techniques to evaluate and verify confidence in their structural integrity and stability. Techniques include root mean square deviation (RMSD), root mean square fluctuation (RMSF), hydrogen bond analysis, and cluster analysis, all available to use as commands through the GROMACS package. RMSD was used to measure the deviations of atom positions within the claudin protein with respect to its reference claudin protein, and RMSF was used to measure these residue deviations over time, to see which residues of the claudin protein experience the most flux. These two techniques indicate the degree of stability between the atoms and the protein structure. As for a hydrogen bond analysis, this measured all the existing hydrogen bonds in the protein's structure and gave an indication of the stability of the secondary structure of the claudin protein. See Figure 7 for the graphs showing the RMSD, RMSF, and hydrogen bond analysis performed for all eight claudins.

In Figure 7A, RMSD shown for each claudin protein exhibits a certain degree of fluctuation, but the RMSF graph in Figure 7B showing spikes at certain residue points may account for the fluctuations observed in the RMSD graphs for their respective claudin proteins. As shown in Figure 7C, the hydrogen bonds for all claudin proteins were generally observed to be occurring at a constant rate, indicating a stable secondary structure. Following these analyses and verifying the stability of the generated claudin structures, a cluster analysis was performed to identify similar structures that were sampled during the MD simulation. One

cluster is chosen from among the others, with being that the chosen cluster contains the biggest similar protein conformations. Following choosing the best cluster, then the setup for self-assembly is completed and the simulations are run.

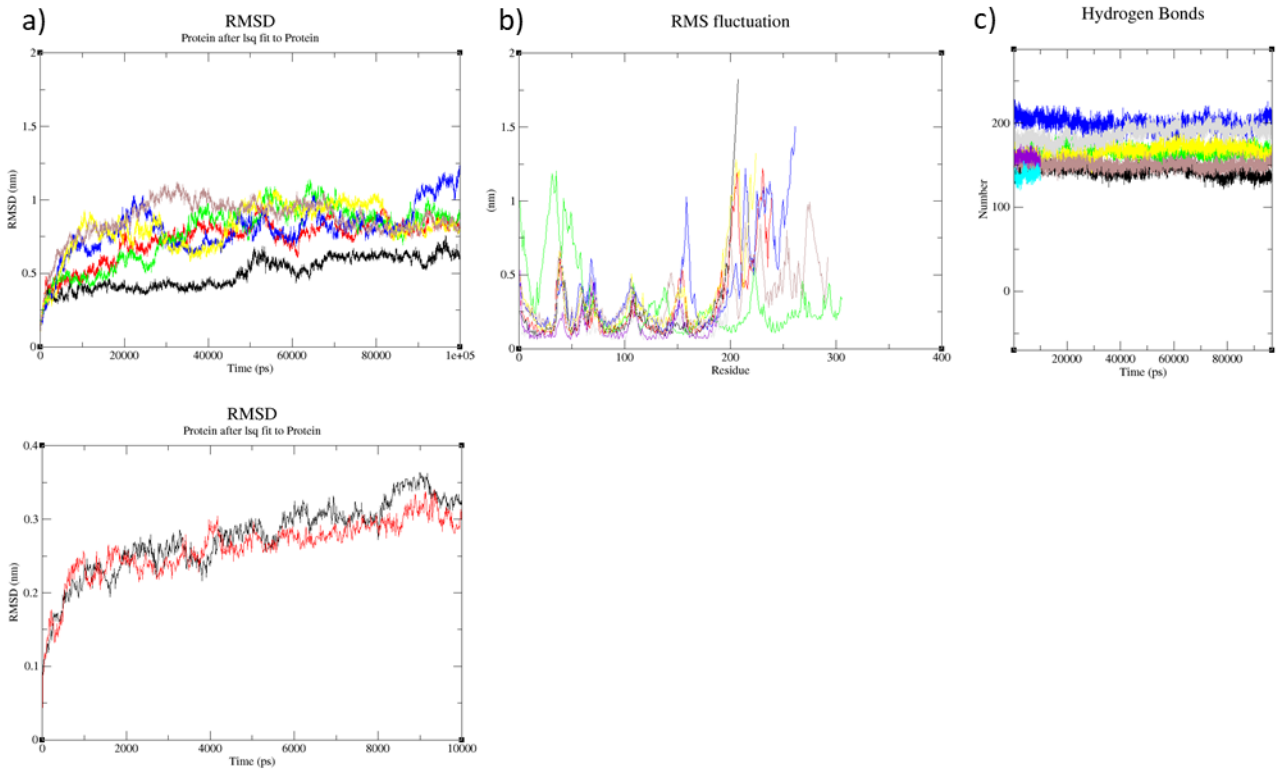


FIGURE 7. *Analysis to evaluate protein structure and stability.* a) RMSD, b) RMSF, c) hydrogen bonds performed on claudin proteins to evaluate confidence in protein structure. a) RMSD graph. Claudins -11, -14, -16, -18, -19, and -23 are shown on the top while claudin -2 and -4 are shown on the bottom half. b) and c) depict RMSF and hydrogen bonding, respectively.

5.2 Self-Assembly Results & Analysis

Upon completion of the ten self-assembly simulations, for the time duration of 4 μs, the following results were observed in Figure 8 and 9 for homotypic and Figure 10 for heterotypic claudin-claudin interactions. After 4 μs, each claudin-claudin interaction, both homotypic and

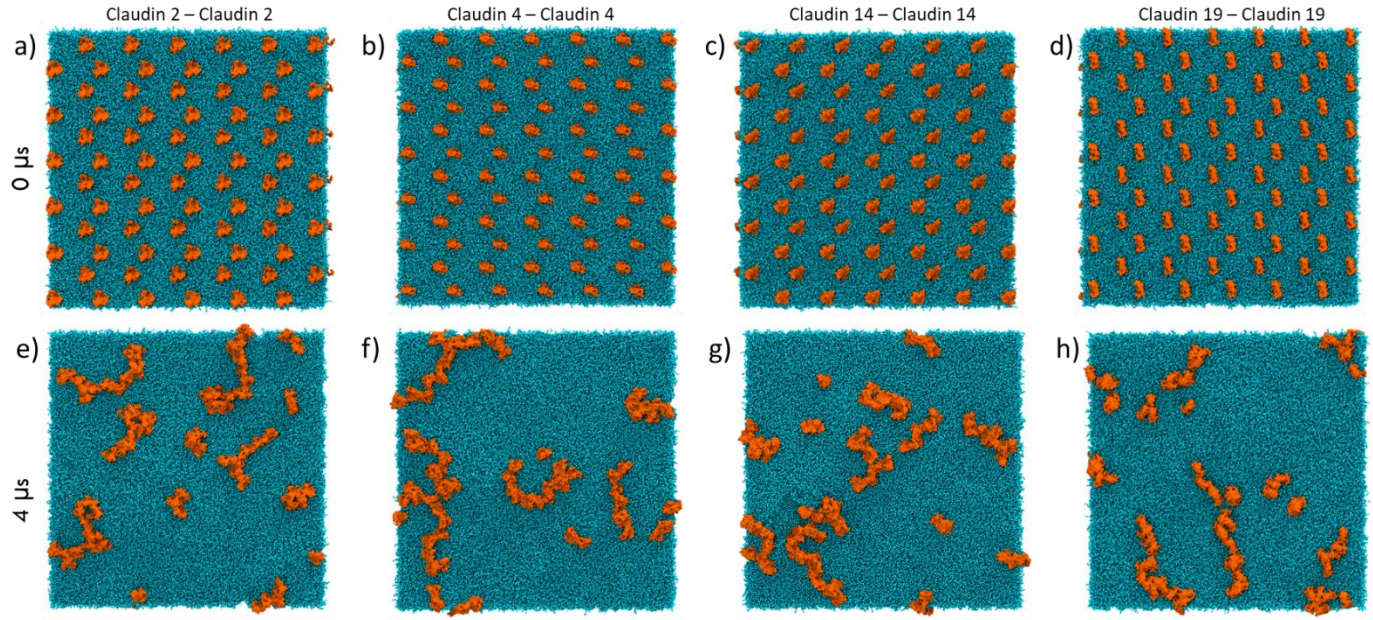


FIGURE 8. *Resulting self-assembly of homotypic classic claudins.* The self-assembly of claudin monomers organizing into strand-like patterns. (A-D) Initial starting configuration of claudin -2/-2, -4/-4, -14/-14, and -19/-19 respectively (left to right) at 0 µs. (E-H) Organized configuration of self-assembled claudin monomers at 4 µs.

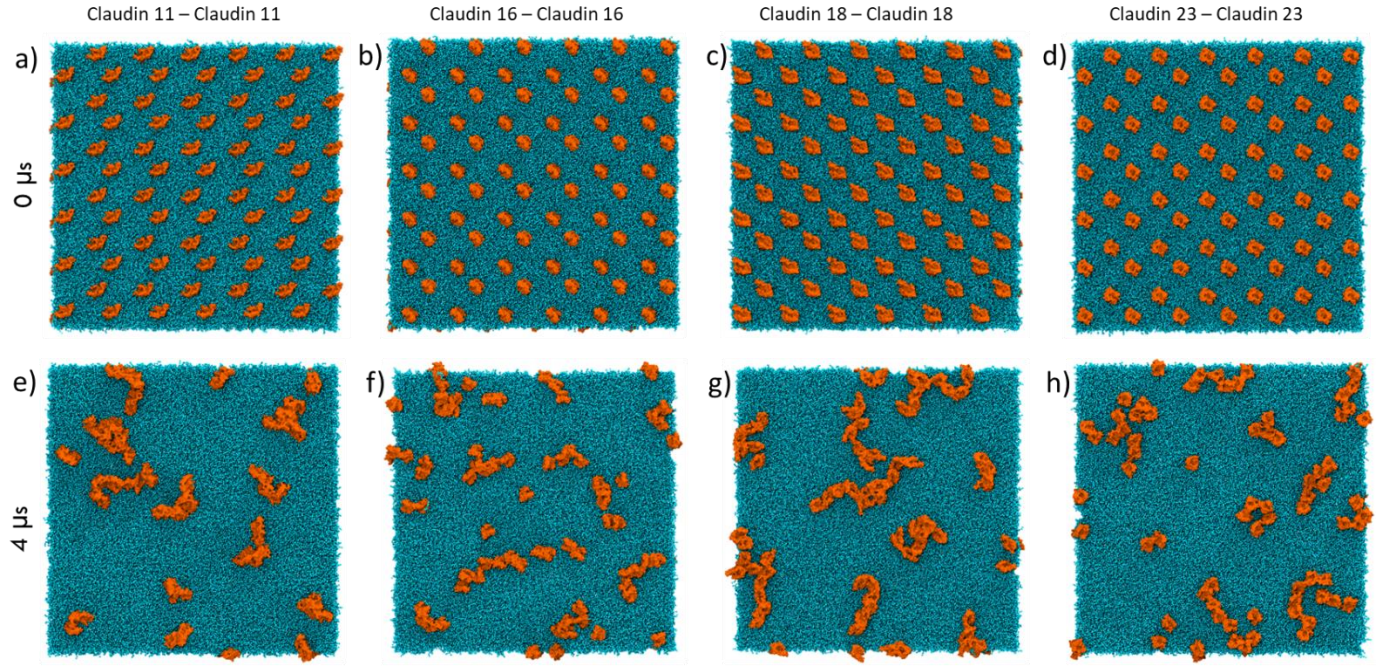


FIGURE 9. *Resulting self-assembly of homotypic non-classic claudins.* Claudin monomers organizing into strand-like patterns. (A-D) Initial configuration at 0 µs of claudin -11/-11, -16/-16, -18/-18, and -23/-23 (left to right), respectively. (E-H) 4 µs configuration of claudins.

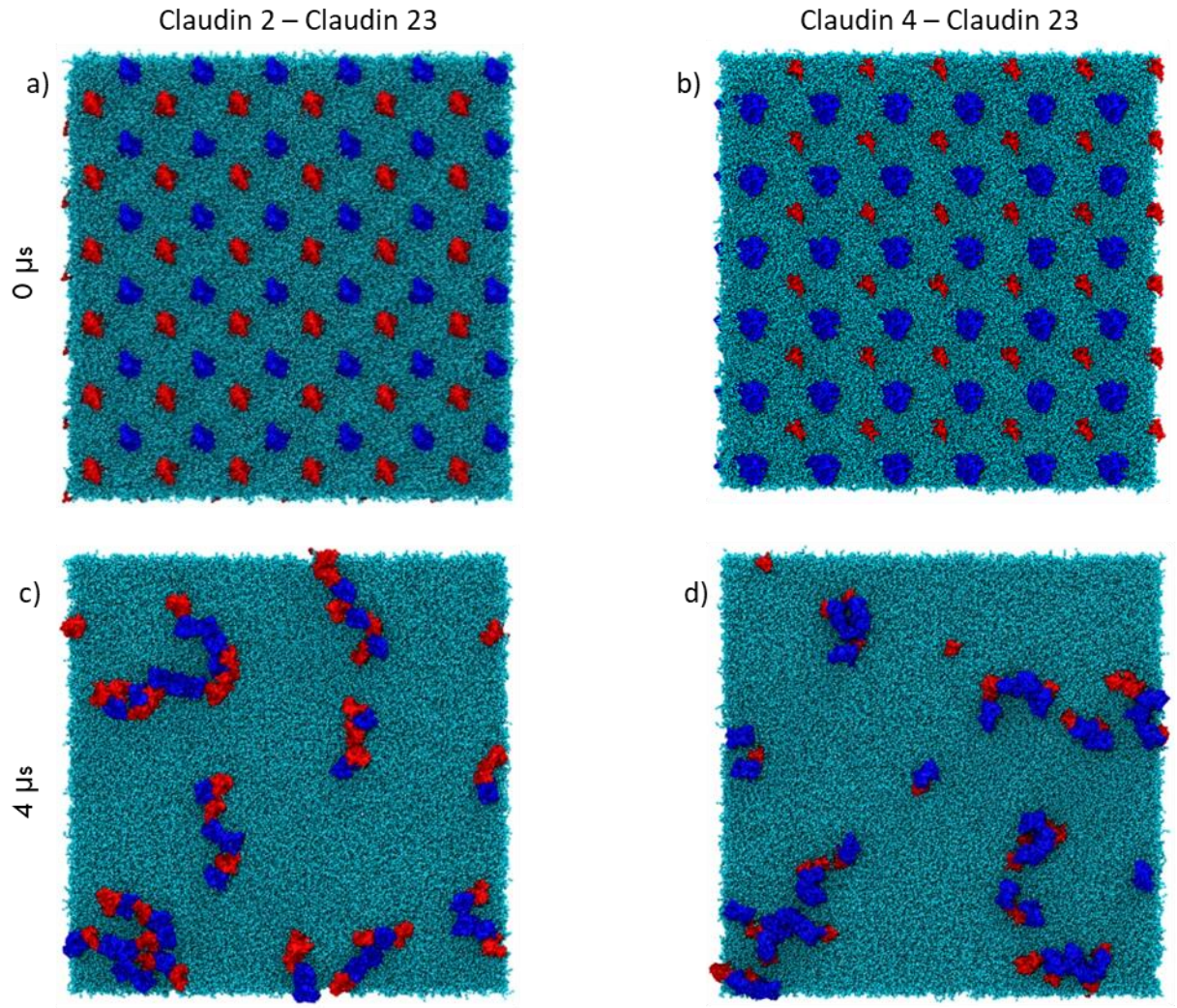


FIGURE 10. *Results from heterotypic self-assembly.* (A-B) 0 µs starting configuration of claudin -2/-23 and -4/-23 (left to right), respectively. (C-D) 4 µs configuration of both interactions, showing dimerization and strand-like organization pattern. Claudin-23 colored red in both left and right, while claudin-2 in blue on left and claudin-4 in blue on right.

heterotypic, self-assembled from an initial starting configuration of isolated monomers into a more organized stand-like assembly, like that which is observed and found in tight junction strands. For each strand-like assembly observed in each claudin-claudin system, the way they have fashioned themselves is contingent on the means of how the claudin monomers within the system have assembled to form dimers and further organize into the resulting strands beheld.

Each interaction system has been analyzed, identifying individual dimer types and their respective density of population, illustrated in Figure 11. From these graphs, it is evident that various claudin-claudin interactions result in different dimer type formations. See Table 4 for an overall comparison of the different dimer types that form from self-assembly as well as their population densities. For the self-assembly dimeric interactions and occurrences between the classic claudins (claudin -2/-2, -4/-4, -14/-14, and -19/-19), since they share a high degree of similarity with each other, one could predict a pattern of dimers to form. However, this was not the observed case, suggesting that regardless of similarity, the individual differences in claudin subtypes result in different dimer type formations. As for the homotypic self-assembly results for the non-classic claudins (claudin -11/-11, -16/-16, -18/-18, and -23/-23), they too do not exhibit any such patterns in forming dimer types. To compare the homotypic interactions of the classic versus the non-classic claudins, the classic claudin self-assemblies were more densely populated in dimer types than their non-classic counterparts.

When looking at the results of the heterotypic systems (claudin -2/-23 and -4/-23), the resulting dimer formations are observed to be slightly different upon interacting with two dissimilar claudin subtypes rather than with two identical subtypes. One case in particular

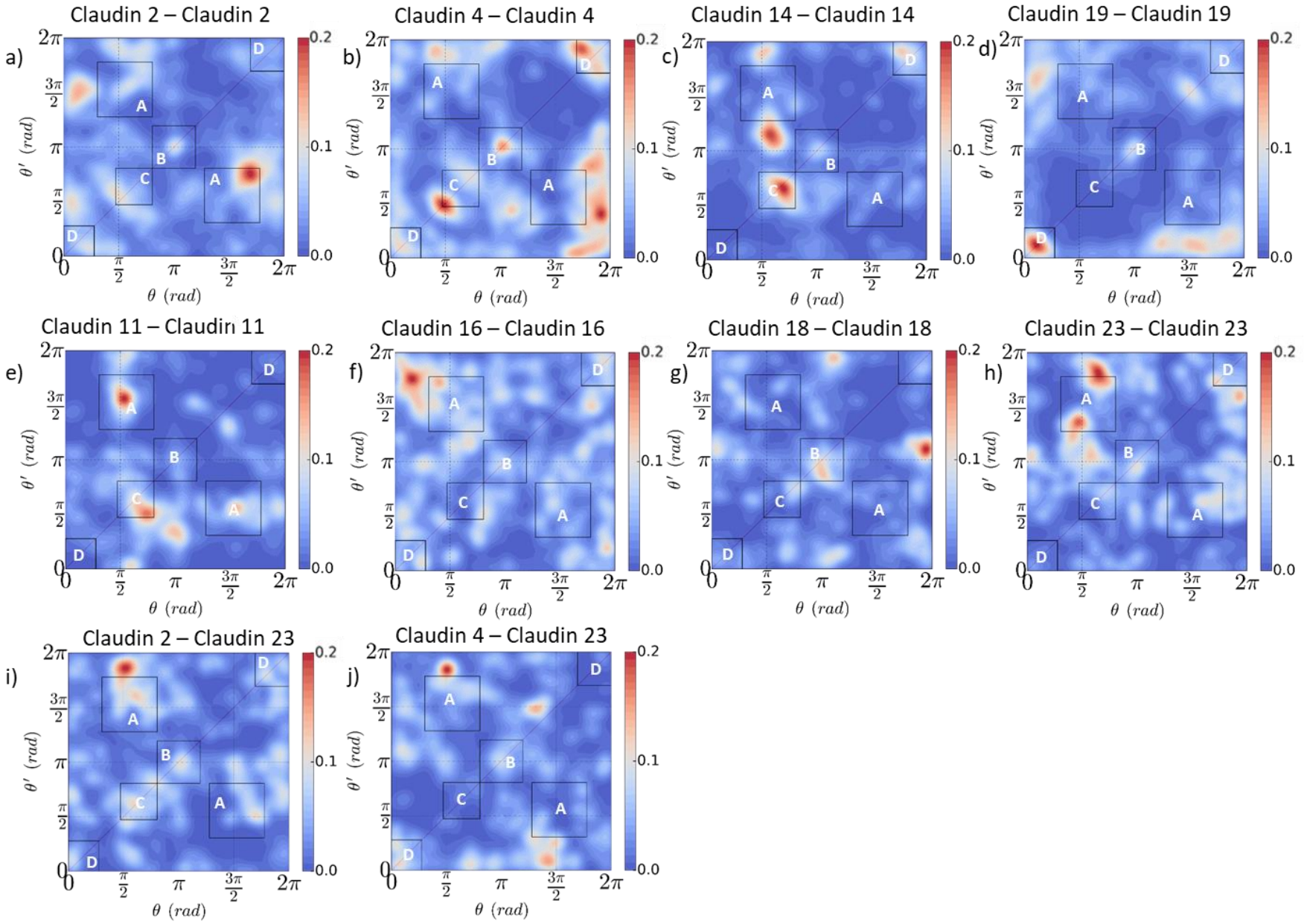


FIGURE 11. *PDF graphs of claudin-claudin self-assembly interactions.* Each graph represents a claudin-claudin self-assembly interaction, identifying dimer types formed as well as probable population densities for each dimer. (A-D) Dimer population and types for homotypic classic claudin interactions. (E-H) Dimer types and populations for homotypic non-classic claudin interactions. (I-J) Dimer types and population density probabilities for heterotypic claudin-claudin interactions.

	Dimer A	Dimer B	Dimer C	Dimer D
Claudin 2 – Claudin 2	+++	+	+	+
Claudin 4 – Claudin 4	+	++	+++	+++
Claudin 14 – Claudin 14	++	+	+++	++
Claudin 19 – Claudin 19	+	+	-	+++
Claudin 11 – Claudin 11	+++	-	+++	-
Claudin 16 – Claudin 16	++	+	++	++
Claudin 18 – Claudin 18	+	++	+	-
Claudin 23 – Claudin 23	+++	++	+	++
Claudin 2 – Claudin 23	+++	++	++	+
Claudin 4 – Claudin 23	++	+	-	++

TABLE 4. *Dimer type formation and population density comparison from self-assembly interactions.* Comprehensive chart translated from respective PDF graph of every individual claudin-claudin interaction, depicting and comparing dimer types and populations. Representation: +++= high density probability (0.15-0.2), ++= medium density (0.1-0.15), += less density (0.5-0.1), X = little to none (0.0-0.5)

regarding the assembly of claudin-4 and claudin-23 together demonstrated an overall lack in formation of Dimer C, as compared to the homotypic assembly interaction of claudin -4/-4 that led to the presence and population of many Dimer C. This result suggests the combination of different claudin subtypes upon assembly interactions will either promote or inhibit the formation of various dimer types, likely due to their distinct individual properties.

5.3 Mapping Energy Landscape Method Results & Analysis

As previously mentioned, this new method of analyzing dimers is still in the refining process. A single test run was computed for claudin-2/-23, sampling 1000 random orientations in the angle space and calculating each energy of interaction. Compared to the sampling done by the self-assembly method, this method was able to sample more of the angle space, illustrated in Figure 12. While the mapping energy landscape method always sampled different dimer orientations each time, the self-assembly method of sampling had overlap with some of the same points.

After generating the random orientations to be sampled, the system undergoes equilibration and a short simulation time of 300 ns for each sample, the resulting energy profile is plotted in a graph, shown in Figure 13. Since the spatial locations of dimers A, B, C, and D are the same as the self-assembly method, the results provide new and additional information regarding the energetic favorability of each dimer conformation. For the claudin-2/-23 interaction, dimer C appears to be energetically favorable, indicated by a lower energy value. The results from the self-assembly method for claudin -2/-23 certainly supports this idea (refer

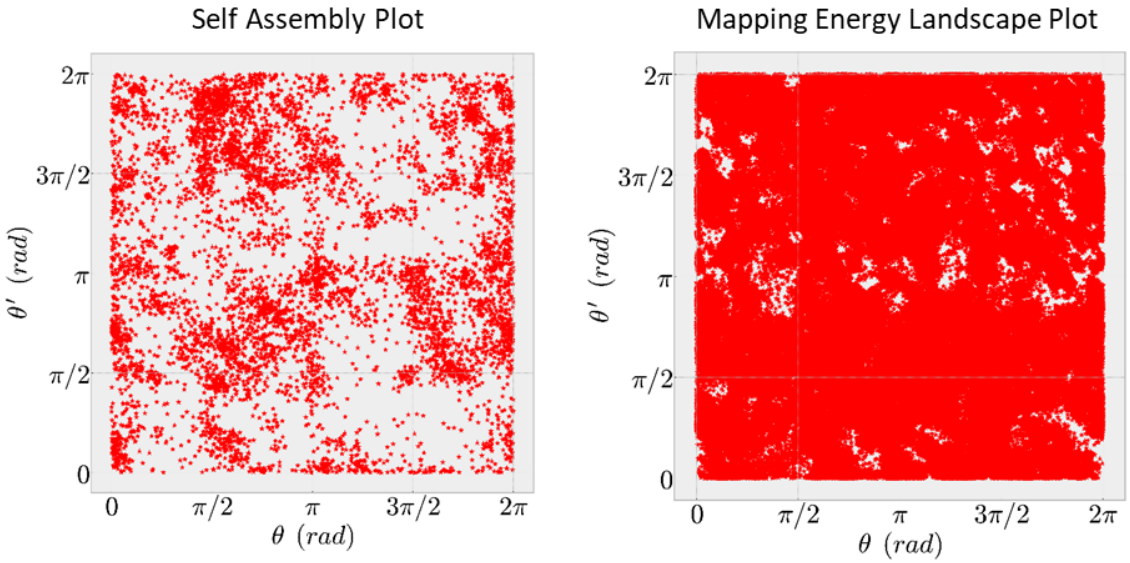


FIGURE 12. *Sampling comparison of self-assembly versus mapping energy landscape plots.* Claudin-2/-23 was analyzed for both plots. Self-assembly did not sample as much of the angle space as the mapping energy landscape plot was able to. Red areas correspond to sampled areas and white areas are unsampled.

to Figure 11I and Table 4). Comparing the two methods, there is a correlation between the two, as areas with a high population in the self-assembly method are seen to have energy dips in the mapping energy landscape method. Certain areas that do not appear to match must be sampled more in the self-assembly method, as this alternative method of mapping energy landscapes should complement the previous density of population method, given enough sampling.

5.4 Comparison of the Two Methods

Both methods of self-assembly and mapping energy landscape analyses provide ways to examine the dimeric interfaces that are essential for claudins to organize into strand-like patterns, as seen in those necessary for establishing tight junction function. Though self-

assembly is a known method used in the Nangia lab, it has its own set of limitations. For one example, the 72 monomer system used to set up the simulation runs for each claudin's self-assembly are very large. One of the self-assembly systems used for this research contained pver 323,964 beads. To process such a large system, using 24 processors/node, the job required weeks (20 days) worth of time for completion, running at a performance rate of 180 ns/day. Additionally, even with the long simulation time and running each system in triplicate, the amount of sampling achieved on the system was never enough. With this new method, the system size is much smaller, thus more affordable and more efficient to compute and run simulations.

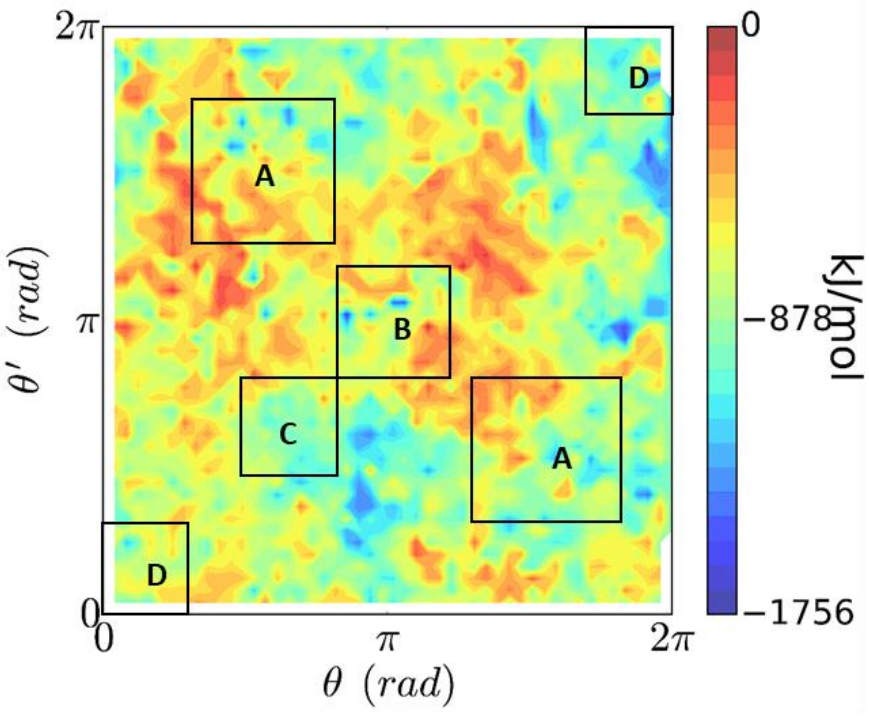


FIGURE 13. *Mapping Energy Landscape energy analysis plot.* Claudin-2/-23 was analyzed. This plot shows energy profiles, an alternative means to showing population density profiles like self-assembly method. Highest energy corresponds to red and lowest energy corresponds to dark blue.

VI. CONCLUSION

As mentioned throughout this thesis, both homotypic and heterotypic systems—classic and non-classic claudins—exhibited a strand-like formation from self-assembly simulations. No one claudin-claudin self-assembly interaction yielded the same results. All the classic claudin-claudin interactions (-2/-2, -4/-4, -14/-14, -19/-19), though highly similar in sequence homology and structure with each other, did not exhibit any specific pattern of dimer formation. All non-classic claudin-claudin interactions (-11/-11, -16/-16, -18/-18, -23/-23) do not demonstrate any patterns of specific dimer types either. Both the homotypic types of interactions formed varying amounts of dimer A, B, C, and D at the end of their self-assembly simulation. The heterotypic interaction was interesting because less of dimer C formed for the heterotypic interaction of claudin -4/-23 compared to the homotypic interaction of claudin -4/-4. This indicates that the pairing of certain claudins with each other may either inhibit the formation of certain dimer types or proliferate the formation of certain dimer types, thereby affecting the overall tight junction network that will resultingly form and subsequently the degree of tight junction permeability. These self-assembly interactions studied were of the *cis* nature, of along the same membrane. The next step should be to see how these same claudin-claudin interactions act in a *trans* interaction and what types of dimers they form, so as to hypothesize which dimers, based on established research, are responsible for a claudin's barrier-forming or pore-forming property.

The structures of the eight claudin proteins being researched in this thesis were not fully known, so various servers and homology modeling were utilized to predict, refine, and construct an accurate 3D claudin structural model to be used for self-assembly simulation. Built in a coarse-grain system, each claudin was put in a lipid bilayer system and energy minimized, equilibrated, and processed before it could be input to a 72 monomer grid to be used for self-assembly simulation. After 4 μs of simulation and observing the results, all the claudin monomers formed strands by creating dimeric interfaces, which were analyzed by using an established dimer analysis method developed by previous colleagues working at the Nangia Lab. When the current method of analyzing self-assembly dimer configurations proved to have its limitations, another method was developed at the Nangia Lab in hopes of addressing and improving upon the previous method. Termed as mapping energy landscapes, this analysis technique makes up for certain flaws that pre-existed in the previous dimer analysis method such as sampling more to make up for the previous inadequate sampling, requiring lesser resources and computational processing time, and providing more qualitative answers in terms of results. If the previous method with self-assembly had enough sampling, the results of the new method should have granted similar results. Since this new method is more efficient, it should be improved upon to be used to better study claudin-claudin interactions. Once all twenty-seven claudins have been studied in all possible combinations, then perhaps this method could be used to study any two interacting transmembrane entities, not limited to just claudin transmembrane proteins.

Comprehensive understanding of the molecular interactions of claudin-claudin proteins is necessary, as since they are key players in controlling tight junction function, they then are promising targets for drug developers to create drugs to treat tight junction-related diseases. Since less is currently known about the homotypic nature of interaction for non-classic claudins as well as certain heterotypic interactions of claudin proteins, it is, therefore, significant to study these interactions, learn about their self-assembly organization, and hypothesize how their orientations that form from self-assembly contribute to overall tight junction barrier function.

VII. APPENDIX

7.1 Scripts to Sample Self-Assembly Population and Analyze Dimer Type Formation

7.1A ANGLE_RW.py script

```
#!/usr/bin/env python
import numpy as np
import argparse
import os, sys
import MDAnalysis
import MDAnalysis.analysis.distances as Mdistance
def angle1 (v1,v2):
# Given two 2D vectors this function will return
#       angles using the atan2 formula.
    return np.math.atan2(np.linalg.det([v1,v2]),np.dot(v1,v2))
def unit_vector(vector):
   """ Returns the unit vector of the vector.  """
   return vector / np.linalg.norm(vector)
n_atom = []
parser = argparse.ArgumentParser()
parser.add_argument('-f',     nargs=1, dest='ref_gro', default=['no'], help=argparse.SUPPRESS,
required=True)
parser.add_argument('-x',     nargs=1, dest='ref_xtc', default=['no'], help=argparse.SUPPRESS,
required=True)
parser.add_argument('-o',     nargs=1, dest='angle_file', default=['angle_file.txt'],
help=argparse.SUPPRESS)
parser.add_argument('-nmol',  nargs=1, dest='n_mol', default=[1], type=int,
help=argparse.SUPPRESS)
parser.add_argument('-tmol',  nargs=1, dest='t_mol', default=[1], type=int,
help=argparse.SUPPRESS)
parser.add_argument('-natom', nargs=1, dest='n_atom', default=[], type=int, action='append',
help=argparse.SUPPRESS)
args = parser.parse_args()
args.ref_gro = args.ref_gro[0]
args.ref_xtc = args.ref_xtc[0]
args.angle_file = args.angle_file[0]
args.n_mol = args.n_mol[0]
args.t_mol = args.t_mol[0]
if not os.path.isfile(args.ref_gro):
   print "Error: file " + str(args.ref_gro) + " not found."
   sys.exit(1)
else:
   args.angle_file = (args.ref_gro[:-4].split('/')[-1]).split('_')[0] + "_angle_file.txt"
if not os.path.isfile(args.ref_xtc):
```

```
    print "Error: file " + str(args.ref_xtc) + " not found."
    sys.exit(1)
if os.path.isfile(args.angle_file):
    print "Error: file " + str(args.angle_file) + " already exisits."
    overwrite = raw_input("Do you wish to overwrite [y/n] : ")
    if overwrite in ['y','Y']:
        angle_file = os.getcwd() + '/' + str(args.angle_file)
        angle_file = args.angle_file
    else:
        sys.exit(1)
else:
    angle_file = os.getcwd() + '/' + str(args.angle_file)
    angle_file = args.angle_file
output = open(angle_file, 'w')
u = MDAnalysis.Universe(args.ref_gro,args.ref_xtc)
if not len(args.n_atom) > 1:
    print "Reading Files"
    num_residues = max(u.atoms.resnums)
    print "Number of Residues", num_residues
    tot_residues = u.select_atoms("name BB").n_atoms
    num_mol      = tot_residues/num_residues
    print "Number of Mol", num_mol
    num_atoms    = u.atoms.n_atoms
    atom_per_mol = num_atoms/num_mol
    print "Number of Atom per mol", atom_per_mol
else:
    mol_natom = []
    if not args.n_mol or not args.t_mol:
        print "please specify the number of molecules and totatl molecules"
        sys.exit(1)
    else:
        print "There are", args.n_mol, " different molecules in the system"
        for i in range(args.n_mol):
            print "mol", i, " has ", args.n_atom[i] ,"atoms"
        for i in range(args.t_mol):
            mol_natom.append(0)
        for i in range(args.t_mol):
            if i % 2 == 0:
                mol_natom[i] = int(args.n_atom[0][0])
            else:
                mol_natom[i] = int(args.n_atom[1][0])
num_frames   = u.trajectory.n_frames
print "Number of trajectories", num_frames
mol         = []
```

```
counter      = 0
counter2     = 0
counter3     = 0
if not len(args.n_atom) > 1:
   mol = [u.atoms[i:i+atom_per_mol] for i in xrange(0,u.atoms.n_atoms,atom_per_mol)]
else:
   j = 0
   for i in xrange(args.t_mol):
      mol.append(u.atoms[j:j+mol_natom[i]])
      j +=  mol_natom[i]
   num_mol =  args.t_mol
dimer_count = 0
time  = 0
for ts in u.trajectory:
   time = time+ts.dt
   print "Processing tracetory frame ts =", time
   for i in range(num_mol):
      for j in range(i+1,num_mol):
         d = Mdistance.distance_array(mol[i].select_atoms("name BB").positions,
mol[j].select_atoms("name BB").positions,box=ts.dimensions, backend='OpenMP')
         if (d <= 10.0).sum() >= 20:
            # C1    = mol[i].select_atoms("name BB").center_of_mass(pbc=True)[0:2]
            # C2    = mol[j].select_atoms("name BB").center_of_mass(pbc=True)[0:2]
            C1    = mol[i].select_atoms("name BB and (resid 12-23 81-93 122-136 165-
180)").center_of_mass(pbc=True)[0:2]
            C2    = mol[j].select_atoms("name BB and (resid 12-23 81-93 122-136 165-
180)").center_of_mass(pbc=True)[0:2]
            X1    = mol[i].select_atoms("name BB and resid 6-26").center_of_mass(pbc=True)[0:2]
            X2    = mol[j].select_atoms("name BB and resid 6-26").center_of_mass(pbc=True)[0:2]
            C1_C2  = C2 - C1
            C2_C1  = C1 - C2
            C1_X1  = X1 - C1
            C2_X2  = X2 - C2
            VC1_X1 = [C1_X1[1],-C1_X1[0]]
            VC2_X2 = [C2_X2[1],-C2_X2[0]]
            alpha_1  = angle1(VC1_X1,C1_C2)
            beta_1   = angle1(C2_X2,C2_C1)
            if beta_1 < 0:
               beta_1 = (2*np.pi)+beta_1
            alpha_2  = angle1(VC2_X2,C2_C1)
            beta_2   = angle1(C1_X1,C1_C2)
            if beta_2 < 0:
               beta_2 = (2*np.pi)+beta_2
            output.write("%.5f \t %.5f\n" % (beta_1,beta_2))
```

```
print "Finished with the Analysis"
output.close()
```

7.1B 2D-KDE.py script

```
#!/usr/bin/env python
import numpy as np
import matplotlib.pyplot as plt
import argparse
import os, sys
plt.style.use('classic')
xmin = 0
xmax = 2*np.pi
Xtick=np.linspace(xmin, xmax, 3,endpoint=True)
Ytick=np.linspace(xmin, xmax, 3,endpoint=True)
x, y = np.mgrid[xmin:xmax:720j, xmin:xmax:720j]
positions = np.vstack([x.ravel(), y.ravel()]).T
from scipy.stats import iqr
d = min(positions.std(),iqr(positions)/1.349)
n = (positions.size/2)**(0.2)
bw =  0.9*(d/n)
def stat_kde(a, b):
   from scipy.stats import gaussian_kde
   ab = np.vstack([a,b])
   knl = gaussian_kde(ab, bw_method='silverman')
   kde = np.reshape(knl(positions).T, x.shape)
   return kde
def sci_kde(a, b):
   from sklearn.neighbors import KernelDensity
   ab = np.vstack([a,b]).T
   knl = KernelDensity(bandwidth=bw, metric='euclidean',
                  kernel='gaussian', algorithm='ball_tree')
   knl.fit(ab)
   kde = np.reshape(np.exp(knl.score_samples(positions)), x.shape)
   return kde
parser = argparse.ArgumentParser()
parser.add_argument('-f',     nargs=1, dest='ang_file', default=['no'], help=argparse.SUPPRESS,
required=True)
args = parser.parse_args()
args.ang_file = args.ang_file[0]
if not os.path.isfile(args.ang_file):
   print "Error: file " + str(args.ang_file) + " not found."
   sys.exit(1)
```

```
else:
   args.image_file = (args.ang_file[:-4].split('/')[-1]).split('_')[0] + "_kde.png"
if os.path.isfile(args.image_file):
   print "Error: file " + str(args.image_file) + " already exisits."
   overwrite = raw_input("Do you wish to overwrite [y/n] : ")
   if overwrite in ['y','Y']:
      image_file = os.getcwd() + '/' + str(args.image_file)
      image_file = args.image_file
   else:
      sys.exit(1)
else:
   image_file = os.getcwd() + '/' + str(args.image_file)
   image_file = args.image_file
data = np.genfromtxt(args.ang_file, dtype=float)
a = data[:,0]
b = data[:,1]
#sk = stat_kde(a,b)
from timeit import default_timer as timer
start = timer()
# ...
ck = sci_kde(a,b)
end = timer()
print(end - start)
fig = plt.figure(figsize=(15,12))
#Set limits
plt.xlim(xmin,xmax)
plt.ylim(xmin,xmax)
plt.plot( [xmin,xmax],[xmin,xmax],'r',
            ls=':', lw=5,
            dash_capstyle='round',
            zorder = ck.min())
levels = np.linspace(ck.min(), ck.max(), 30)
kplot  = plt.contourf(x, y, ck,
                levels=levels,
                cmap=plt.cm.coolwarm,
                extent=[xmin, xmax, xmin, xmax],
                norm=plt.Normalize(vmax=abs(ck).max(), vmin=abs(ck).min()),
                alpha=0.90)
if ck.max() > 0.16:
   cbar = fig.colorbar(kplot,format="%.1f")
else:
   cbar = fig.colorbar(kplot,format="%.2f")
cbar.ax.tick_params(length=10,width=3,labelsize=40)
cbar.set_ticks(np.linspace(ck.min(),ck.max(),3))
```

```
plt.xticks(Xtick,[r'$0$',r'$\pi$',r'$2\pi$'],fontsize=50)
plt.yticks(Ytick,[r'$0$',r'$\pi$',r'$2\pi$'],fontsize=50)
plt.xlabel(r'$\theta \/\ (rad)$',fontsize=50,weight='bold')
plt.ylabel(r'$\theta ^\prime \/\ (rad)$',fontsize=50,weight='bold')
# Draw contour lines
#cbar.set_label(r'PDF',fontsize=50)
plt.tight_layout()
plt.savefig(args.image_file, transparent=True, bbox_inches='tight',dpi=300)
#plt.show()
```

VIII. REFERENCES

1. Cereijido, M., Contreras, R. G. & Shoshani, L. Cell adhesion, polarity, and epithelia in the dawn of metazoans. Physiol. Rev. 84, 1229–1262 (2004).
2. Farquhar, M. G. & Palade, G. E. Junctional complexes in various epithelia. J. Cell Biol. 17, 375–412 (1963).
3. Van Itallie, C. M. & Anderson, J. M. The Molecular Physiology of Tight Junction Pores The Molecular Physiology of Tight. Physiology 19, 331–338 (2004).
4. Zihni, C., Mills, C., Matter, K. & Balda, M. S. Tight junctions: From simple barriers to multifunctional molecular gates. Nat. Rev. Mol. Cell Biol. 17, 564–580 (2016).
5. Claude, P. & Goodenough, D. A. Fracture faces of zonulae occludentes from “tight” and “leaky” epithelia. J. Cell Biol. 58, 390–400 (1973).
6. Staehelin, L. A., Mukherjee, T. M. & Williams, A. W. Freeze-etch appearance of the tight junctions in the epithelium of small and large intestine of mice. Protoplasma 67, 165–184 (1969).
7. Furuse, M. et al. Overexpression of occludin, a tight junction integral membrane protein, induces the formation of intracellular multilamellar bodies bearing tight junction-like structures. J. Cell Sci. 109, 429–435 (1996).
8. Furuse, M., Sasaki, H., Fujimoto, K. & Tsukita, S. A single gene product, claudin-1 or -2, reconstitutes tight junction strands and recruits occludin in fibroblasts. J. Cell Biol. 143, 391–401 (1998).

9. Morita, K., Furuse, M., Fujimoto, K. & Tsukita, S. Claudin multigene family encoding four-transmembrane domain protein components of tight junction strands. Proc. Natl Acad. Sci. USA 96, 511–516 (1999).
10. Kubota, K. et al. Ca2+ -independent cell-adhesion activity of claudins, a family of integral membrane proteins localized at tight junctions. Curr. Biol. 9, 1035–1038 (1999).
11. Osler, M. E., Chang, M. S. & Bader, D. M. Bves modulates epithelial integrity through an interaction at the tight junction. J. Cell Sci. 118, 4667–4678 (2005).
12. Luissint, A. C., Nusrat, A. & Parkos, C. A. JAM-related proteins in mucosal homeostasis and inflammation. Semin. Immunopathol. 36, 211–226 (2014).
13. Martin-Padura, I. et al. Junctional adhesion molecule, a novel member of the immunoglobulin superfamily that distributes at intercellular junctions and modulates monocyte transmigration. J. Cell Biol. 142, 117–127 (1998).
14. Cohen, C. J. et al. The coxsackievirus and adenovirus≈receptor is a transmembrane component of the tight junction. Proc. Natl Acad. Sci. USA 98, 15191–15196 (2001).
15. Higashi, T. et al. Analysis of the 'angulin' proteins LSR, ILDR1 and ILDR2—tricellulin recruitment, epithelial barrier function and implication in deafness pathogenesis. J. Cell Sci. 126, 966–977 (2013).
16. Masuda, S. et al. LSR defines cell corners for tricellular tight junction formation in epithelial cells. J. Cell Sci. 124, 548–555 (2011).
17. Yu, A. S. Claudins and the kidney. J. Am. Soc. Nephrol. 26, 11–19 (2015).

18. Yu, A. S. et al. Molecular basis for cation selectivity in claudin-2-based paracellular pores: identification of an electrostatic interaction site. J. Gen. Physiol. 133, 111–127 (2009).
19. Lingaraju, A. et al. Conceptual barriers to understanding physical barriers. Semin. Cell Dev. Biol. 42, 13–21 (2015).
20. Rajasekaran, A. K., Hojo, M., Huima, T. & Rodriguez-Boulan, E. Catenins and zonula occludens-1 form a complex during early stages in the assembly of tight junctions. J. Cell Biol. 132, 451–463 (1996).
21. Maiers, J. L., Peng, X., Fanning, A. S. & DeMali, K. A. ZO-1 recruitment to α-catenin — a novel mechanism for coupling the assembly of tight junctions to adherens junctions. J. Cell Sci. 126, 3904–3915 (2013).
22. Fukuhara, A. et al. Involvement of nectin in the localization of junctional adhesion molecule at tight junctions. Oncogene 21, 7642–7655 (2002).
23. Zihni, C., Balda, M. S. & Matter, K. Signalling at tight junctions during epithelial differentiation and microbial pathogenesis. J. Cell Sci. 127, 3401–3413 (2014).
24. Gonzalez-Mariscal, L. et al. Tight junctions and the regulation of gene expression. Semin. Cell Dev. Biol. 36, 213–223 (2014).
25. Quiros, M. & Nusrat, A. RhoGTPases, actomyosin signaling and regulation of the epithelial apical junctional complex. Semin. Cell Dev. Biol. 36, 194–203 (2014).
26. Balda, M. S. et al. Assembly and sealing of tight junctions: possible participation of G-proteins, phospholipase C, protein kinase C and calmodulin. J. Mem. Biol. 122, 193–202 (1991).

27. Garrido-Urbani, S., Bradfield, P. F. & Imhof, B. A. Tight junction dynamics: the role of junctional adhesion molecules (JAMs). Cell Tissue Res. 355, 701–715 (2014).
28. Powell DW. Barrier function of epithelia. Am J Physiol Gastrointest Liver Physiol 241: G275–G288, (1981).
29. Tsukita S and Furuse M. Claudin-based barrier in simple and stratified cellular sheets. Curr Opin Cell Biol 14: 531–536, (2002).
30. Sawada, N. *et al.* Tight junctions and human diseases. *Med. Electron Microsc.* 36, 147–156 (2003).
31. Furuse, M. *et al.* Occludin : A Novel Integral Membrane Protein Localizing at Tight Junctions. *J. Cell Biol.* 123, 1777–1788 (1993).
32. Furuse, M. (2010). Chapter 1 - Introduction: Claudins, Tight Junctions, and the Paracellular Barrier. Current Topics in Membranes. A. S. L. Yu, Academic Press. 65: 1-19.
33. Gunzel, D. & Yu, A. S. L. *Claudins and the Modulation of Tight Junction Permeability*. *Physiological Reviews* 93, (2013).
34. Krause, G. *et al.* Structure and function of claudins. *Biochim. Biophys. Acta - Biomembr.* 1778, 631–645 (2008).
35. Haseloff, R. F., Piontek, J. & Blasig, I. E. The Investigation of cis- and trans-Interactions Between Claudins. *Curr. Top. Membr.* 65, 97–112 (2010).
36. Müller, D. *et al.* A novel claudin 16 mutation associated with childhood hypercalciuria abolishes binding to ZO-1 and results in lysosomal mistargeting. *Am. J. Hum. Genet.* 73, 1293–1301 (2003).

37. Van Itallie, C. M. & Anderson, J. M. Claudin interactions in and out of the tight junction. *Tissue Barriers* 1, e25247 (2013).

38. Furuse, M. *et al.* Claudin-based tight junctions are crucial for the mammalian epidermal barrier: A lesson from claudin-1-deficient mice. *J. Cell Biol.* 156, 1099–1111 (2002).

39. D'Souza, T., Agarwal, R. & Morin, P. J. Phosphorylation of Claudin-3 at threonine 192 by cAMP-dependent protein kinase regulates tight junction barrier function in ovarian cancer cells. *J. Biol. Chem.* 280, 26233–26240 (2005).

40. Hou, J. *et al.* Claudin-16 and claudin-19 interact and form a cation-selective tight junction complex. *J. Clin. Invest.* 118, 619–628 (2008).

41. Milatz, S. *et al.* Tight junction strand formation by claudin-10 isoforms and claudin-10a/-10b chimeras. *Ann. N. Y. Acad. Sci.* 1405, 102–115 (2017).

42. Irudayanathan, F. J., Trasatti, J. P., Karande, P. & Nangia, S. Molecular Architecture of the Blood Brain Barrier Tight Junction Proteins-A Synergistic Computational and in Vitro Approach. *J. Phys. Chem. B* 120, 77–88 (2016).

43. Irudayanathan, F. J., Wang, N., Wang, X. & Nangia, S. Architecture of the paracellular channels formed by claudins of the blood–brain barrier tight junctions. *Ann. N. Y. Acad. Sci.* 1405, 131–146 (2017).

44. Irudayanathan, F. J. *et al.* Self-assembly Simulations of Classic Claudins-Insights into the Pore Structure, Selectivity and Higher Order Complexes. (2018).

45. Whitesides, G. M. & Grzybowski, B. Self-assembly at all scales. *Science (80-.).* 295, 2418–2421 (2002).

46. Durrant, J. & McCammon, J. A. Molecular dynamics simulations and drug discovery. *BMC Biol.* 9, 1–9 (2011).

47. Jo, S., Kim, T., Iyer, V. G. & Im, W. CHARMM-GUI: A Web-Based Graphical User Interface for CHARMM. *J. Comput. Chem.* 29, 1859–1865 (2008).

48. Hsu, P. C., Jefferies, D. & Khalid, S. Molecular Dynamics Simulations Predict the Pathways via Which Pristine Fullerenes Penetrate Bacterial Membranes. *J. Phys. Chem. B* 120, 11170–11179 (2016).

49. Berendsen, H. J. C., van der Spoel, D. & van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.* 91, 43–56 (1995).

50. Zhang, Y. I-TASSER server for protein 3D structure prediction. *BMC Bioinformatics* 9, 1–8 (2008).

51. Xu, D. & Zhang, Y. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins Struct. Funct. Bioinforma.* 80, 1715–1735 (2012).

52. Zhang, J., Liang, Y. & Zhang, Y. Atomic-level protein structure refinement using fragment-guided molecular dynamics conformation sampling. *Structure* 19, 1784–1795 (2011).

53. Feig, M. Local Protein Structure Refinement via Molecular Dynamics Simulations with locPREFMD. *J. Chem. Inf. Model.* 56, 1304–1312 (2016).

54. Ma, J., Wang, S., Wang, Z. & Xu, J. Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning. *Bioinformatics* 31, 3506–3513 (2015).

55. López-Blanco, J. R., Canosa-Valls, A. J., Li, Y. & Chacón, P. RCD+: Fast loop modeling server. *Nucleic Acids Res.* 44, W395–W400 (2016).

56. Lomize, M. A., Pogozheva, I. D., Joo, H., Mosberg, H. I. & Lomize, A. L. OPM database and PPM web server: Resources for positioning of proteins in membranes. *Nucleic Acids Res.* 40, 370–376 (2012).

57. Krieger, E. & Vriend, G. New ways to boost molecular dynamics simulations. *J. Comput. Chem.* 36, 996–1007 (2015).

58. Bateman, A. *et al.* UniProt: The universal protein knowledgebase. *Nucleic Acids Res.* 45, D158–D169 (2017).

59. Shinoda, T., et al. (2016). "Structural basis for disruption of claudin assembly in tight junctions by an enterotoxin." Sci Rep 6: 33632.

60. Suzuki, H., et al. (2014). "Crystal structure of a claudin provides insight into the architecture of tight junctions." Science 344(6181): 304-307.

61. Saitoh, Y., et al. (2015). "Tight junctions. Structural insight into tight junction disassembly by Clostridium perfringens enterotoxin." Science 347(6223): 775-778.

62. Wassenaar, T. A., Pluhackova, K., Böckmann, R. A., Marrink, S. J. & Tieleman, D. P. Going backward: A flexible geometric approach to reverse transformation from coarse grained to atomistic models. *J. Chem. Theory Comput.* 10, 676–690 (2014).

63. De Jong, D. H. *et al.* Improved parameters for the martini coarse-grained protein force field. *J. Chem. Theory Comput.* 9, 687–697 (2013).

64. Wassenaar, T. A., Ingólfsson, H. I., Böckmann, R. A., Tieleman, D. P. & Marrink, S. J. Computational lipidomics with insane: A versatile tool for generating custom membranes for molecular simulations. *J. Chem. Theory Comput.* 11, 2144–2155 (2015).
65. Humphrey, W., Dalke, A. & Schulten, K. {}: {Visual} molecular dynamics. *J. Mol. Graph.* 14, 33–38 (1996).

IX. VITA

Author Name: Lisa Nguyen

Date of Birth: October 21, 1994

Undergraduate School Attended:

MCPHS University (2012-2016)

Degrees Awarded:

B.S. Medical and Molecular Biology, May 2016